ISO TC 184/SC4/WG2 **ISO CD 13584-10**
Date: February 24th, 1995

INDUSTRIAL AUTOMATION SYSTEMS AND INTEGRATION

PARTS LIBRARY

Part 10 Title: Conceptual Model of Parts Library

ABSTRACT: This Conceptual Model document presents the problem domain analysis of the field covered by the Parts Library Standard. It defines the concepts, the terminology and the mechanisms for parts library information representation and exchange. It specifies the requirements for standardisation.

CD version

Editor: Pr. G. Pierra
Address: Ecole Nationale Supérieure de Mécanique et d'Aérotechnique
Laboratoire d' Informatique Scientifique et Industrielle
Site du Futoroscope - B.P. 109
F-86960 Futoroscope CEDEX
France

Tele: +33 49 49 80 60
Fax +33 49 49 80 64
E-mail pierra@ensma.univ-poitiers.fr

Content

FIGURES

Foreword

ISO (the International Organisation for Standardisation) is a world-wide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International-organisations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardisation.

Draft International Standards adopted by technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval-by at least 75% of the member bodies casting a vote.

International Standard ISO 13584-10 was prepared by *Technical Committee ISO 184, Industrial automation systems and integration*, Subcommittee *SC 4, Industrial data and global manufacturing programming languages.*

This is the first edition of this part of ISO 13584.

There are three annexes to this part. All are informative.

The preparation of this document has benefitted from the contribution of the PLUS project (ESPRIT III 8984), partly funded by the Commission of European Community.

ISO 13584 consists of the following parts under the general title Industrial automation systems and integration - Parts Library:

Part 1 Overview and fundamental principles[1]

Part 20 General resources[1]

Part 24 Logical model of supplier library[1]

Part 26 Identification of library suppliers[1]

Part 31 Programming interface[1]

Part 42 Methodology for structuring parts families[1]

Part 101 Geometric view exchange protocol by parametric program[1]

Part 102 Geometric view exchange protocol by ISO 10303 conforming specification[1]

Introduction

Products are often made of parts. Therefore, the information generated about a product during its design, manufacture, utilisation, maintenance and disposal contains the information about the parts the product consists of. When the parts in a product are defined independently of the product itself, e.g. by a part supplier, the information about them is independent of any product where they may be used. This information may be exchanged between different organisations to facilitate the generation of information about products that contain such parts.

ISO 13584 is a series of International Standards for the computer-readable representation and exchange of parts library data. The objective is to provide a mechanism capable of transferring parts library data, independent of any application which is using a parts library data system. The nature of this description makes it suitable not only for the exchange of files containing parts, but also as a basis for implementing and sharing databases of parts library data.

Each International Standard in the ISO 13584 series is published as a separate part. Parts are grouped into one of the following series: conceptual descriptions (reserved part numbers are 10 to 19), logical resources (20 to 29), implementation resources (30 to 39), description methodology (40 to 49), conformance testing (50-59), view exchange _____

[1]To be published

protocol (101 to 199) protocol and standardised content (2000 to 2999). The numbers are for unambiguous reference to the documents.

This PART of ISO 13584 belongs to the conceptual description series of PARTs. It presents a problem domain analysis of the field covered by this International Standard. It specifies the requirements for standardisation. It defines precise concepts, terminology and mechanisms for this International Standard. The other PARTs of ISO 13584 are as follows:

ISO 13584-1 Overview and fundamental principles[1];

ISO 13584-20 General resources[1];

ISO 13584-24 Logical model of supplier library[1];

ISO 13584-26 Identification of library suppliers[1];

ISO 13584-31 Programming interface[1];

ISO 13584-42 Methodology for structuring parts families[1] ;

ISO 13584-101 Geometric view exchange protocol by parametric program[1];

ISO 13584-102 Geometric view exchange protocol by ISO 10303 conforming specification[1].

There are three annexes to this part; all of them are informative.

[1]To be published

Industrial automation systems and integration-
Parts Library-
Part 10:
Conceptual descriptions: Conceptual Model of Parts Library

1 Scope

The ISO 13584 series provides a representation of parts library information together with the necessary mechanisms and definitions to enable parts library data to be exchanged, used and updated. The exchange will be between different computer systems and environments associated with the complete life cycle of the products where the library parts may be used, including product design, manufacture, utilisation, maintenance, and disposal.

This part of ISO 13584 presents the object oriented analysis of the problem domain and provides the conceptual model which constitutes the conceptual basis of this International Standard. It defines the basic concepts and terminology about representation and exchange of parts library information. It also specifies the requirements for standardisation.

The following are within the scope of this part of ISO 13584:

– a problem domain analysis of the information involved in parts library representation and exchange;

– the description of reference models, i.e., models of parts libraries systems seen from different points of view:
- The view of the roles involved,
- The view of the system architecture,
- The view of the information structure,
- The view of the information semantic.

– the requirements for standardisation;

The following are outside the scope of this part of ISO 13584:

– the resources used for parts library information representation and exchange;

– the information models used in this International Standard;

2 Normative references

This part of ISO 13584 incorporates by dated or undated reference, provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this part of ISO 13584 only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies.

ISO 1539: 1991	*Information technology; Programming languages; FORTRAN*
ISO 4014: 1988	*Hexagon head bolts - Product grades A and B*
ISO 8879: 1986	*Information processing ; Text and office systems; Standard Generalised Mark-up Language (SGML)*
ISO 10303-11: 1994	*Industrial automation systems: Product data representation and exchange - Part 11: Description Methods: The EXPRESS language reference manual*
ISO 10303-21: 1994	*Industrial automation systems: Product data representation and exchange - Part 21: Implementation methods: Clear text encoding of the exchange structure*
ISO 10303-43: 1994	*Industrial automation systems: Product data representation and exchange - Part 43: Integrated recources: Representation structures*

3 Definitions and abbreviations

An extended terminology index of the set of concepts which are used in more than one part of this International Standard for parts library information representation and exchange is presented below. This set of extended definitions, which includes and extends those given in ISO 13584-1, is mainly provided for reference purposes. The definitions are given in alphabetical, not logical order. Hence, they use forward references. They are not intended to be read sequentially, but to provide more accurate definitions of the concepts introduced in clause 4 of this part of ISO 13584.

The extended definitions given here are not general, but define the particular meaning of the different terms in the context of the Parts Library Standard. As far as possible, the same terminology as ISO 10303 is used.

In order to cover the range of this International Standard, this index supplies the definitions for three classes of concepts:
1) the concepts that are within the scope of the problem domain (e.g., family of parts)
2) the terms that are relevant to the information model used to represent them in a library (e.g., class of general models)
3) the terms that are relevant to the information model used to represent them in a CAD system database (e.g., part occurrence).

A table below outlines the three conceptual levels.

In the extended definitions of the concepts the items in italics refer to terms that are also defined.

3.1 Outline of terminology

Table 1, below, is designed to facilitate the understanding of the terminology index, and summarises a few basic corresponding points among the three levels. Some of the terms that are common within the problem domain do not have a very precise meaning (e.g., parameter). Hence, their use is avoided in this International Standard. They are included in the table below only to facilitate understanding.

Problem terminology	**Model in a library**	**Model in an end-user CAD system or in an exchange between CAD systems**
Simple families of parts	*General model classes*	
Generic families of parts: an abstraction of different simple families of parts, e.g., the screws, the circular bearings, the bearings	non leaf node of a *general model class* hierarchy	instances are *generic parts*
hierarchical classification of *families* with (possible) factorisation of common properties	hierarchy of *general model classes with* (possible) *inheritance*	
a "specimen" part. Covers two concepts: • an abstraction where only the characteristics are defined (e.g.: Hex screw 10 x 5 ...) = part *instance* • a precise part incorporated and positioned in a given product = *occurrence* of a part *instance*	• *General model (instance* of a *general model class)* • no representation	• *Identifier* (will be contained in a *general view)* • *General view: object_name* (represents the *identifier* + a notion of position)
representations (E.g.: geometry, symbol...)	*Functional models* (grouped into *classes* associated with *general model classes*)	*Functional views*
representation category (perspective of the user)	*functional view class* associated with*:* • *view logical names* • values of *view control variables (E.g.: Geometry solid standard)*	*view logical name* is the name of the view; *view control variables* are *attributes* of a view they both characterise the user perceptive
content of the representation (e.g., set of geometric entities)	method (e.g., program) triggered by a message whose: name = *view logical name* parameters = *view control variables*	*functional view*: SUBTYPE of ISO 10303-43 representation + set of *view attributes*
simple family of standardised *parts* (describes an "abstract" population, that consists of all the supplier parts conforming to the standard)	*general model class* of *abstract parts*	
simple family of *suppliers parts* (describes a concrete population)	*general model class* of *supplier parts*	
the *supplier family* conforms to some standard (the concrete population belongs to the abstract population)	*class* of *general models* that are *case_of* the other *class*	the *general view* of the *supplier part* may be associated by a *case_of* relationship with a *general view* of the *abstract part*
"parameter" (property that relates to a part or to a representation)	*part attribute, context parameter, behaviour representation attribute, representation attribute*	
identification parameter	*identification attribute*	constituent field of the *identifier*
parameter not used for identification	*derived attribute* (a function necessarily exists for inferring them from the identification attributes)	
(family, parameter) name	*code* and *version* of the *dictionary* corresponding entry (associated with external language-dependant representations)	*absolute identifier* (+ possibly external representations)
identifier	*supplier code* + *general model class code* and *version* + list of: (values of *identification attributes*)	character strings(s) used to represent this content (according to formats)

Table 1: Outline of terminology

3.2 Definitions

For the purpose of the ISO 13584 part series, the following definitions apply.

• **Absolute identifier**: every entry included in the *semantic dictionary* that describes a *library* has an absolute identifier that does not vary over time and is language-independent. This name is structured and is constructed from the *codes* and *versions* of the *semantic dictionary* entries, in the following form:
- supplier absolute identifier = supplier_code-000;
- absolute identifier of a *class* = supplier absolute identifier•class_code-class version;
- Absolute identifier of an item in a *class* (e.g. attribute) = class absolute identifier•element code-element version.

EXAMPLE 1: FR00001-000.E25112-003.D-002 (diameters of hex screws belonging to the E25112 family defined by AFNOR)

• **Abstract part:** a specification that represents, at some level of abstraction, different *supplier parts. An abstract* part belongs to a *family of parts* defined by a partial specification (e.g., international standard, company standard, function specification). The family of parts that defines such an element is described by its *supplier* as a *class* (compare with: *supplier part, generic part*).

EXAMPLE 2: The concept of *abstract part* allows the instanciation of an "ISO 4014 bolt" within a design without caring about the *supplier part* that will be finally used in the product. This capability requires the availability of some functional model (e.g. geometry) of such an instance.

• **Applicable property**: a *property* that shall apply to any part that belongs to a *family of parts*, or to any of its *descendent* family. Applicable properties are a subset of the *visible properties.* Applicable properties are *inherited.*

• **Ascendant of a family**: X is the (immediate) ascendant family of a *family* Y, if the *is_a* relationship (inheritance) exists between them, i.e. Y is_a X. Then an *ascendant family* of (some family) A is either the immediate ascendant of A or an ascendant of the immediate ascendant of A.

• **Assembled part**: a *part* that may be decomposed into a set of *components* or other assembled parts. It can only be represented in a *level* 2 library. There are two types of assembled part, which differ in the way in which the assembled part is *identified*: *structured parts* and *assemblies.*

• **Assembly**: *assembled part* that has no *identification* of its own, but is identified by its constituent parts. The *identification attributes* of an assembly are therefore *identifiers* of its constituent objects. The representation of a set of similar assemblies in the form of a *general model class*, (1) enables assembly constraints to be expressed (these constraints may be used by the system to guide the user in his selection process) and (2) allows a *functional-model class* to be associated with such a class to produce *functional views* of the assembly.

EXAMPLE 3: A nut-and-bolt assembly is identified by the bolt and the nut that constitute it. The type of the bolt is defined by its family class (simple or generic). The same applies for the type of the nut. The description of the nut-and-bolt assembly class therefore (1) enables the class to be specified from its constituent items and (2) allows for the expression of certain integrity constraints necessary for an allowable assembly. For example, the equality constraint between the bolt thread diameter and the nut thread diameter may be used by the system to propose only compatible nuts once the screw has been chosen. Similarly, a functional geometrical model of the assembly will produce a correct 2-dimensional drawing of the assembly, whereas a solid kinematics functional model will produce two solids, in association with a prismatic joint.

• **Attribute**: generic term used to characterise any property of any kind of entity that belongs either within the area of the problem (the universe of parts) or to its computer model (the object oriented model presented in this International Standard). Each category of attribute used in this standard possesses its own definition. In a computerised library, every attribute is defined within a class (e.g., the *general model class*, the *functional model class*, the *functional view class*). It possesses a code assigned by the supplier of the class. That code shall be unique for the class and within all its inheritor subclasses. It possesses a *version.* It also possesses a *descriptor* (see: *derived attribute, free attribute, generic attribute, identification attribute, inherited attribute, instance attribute, representation attribute, view attribute, behaviour representation attribute*).

• **Attribute of a part (Level 1)**: an invariable property, characteristic of a *part*, whose value, can be inferred from the part *identifier* once the part is chosen, regardless of the context in which the part is inserted. The type of an attribute may be numeric, alphanumeric or Boolean.

To produce a *functional view* of a part, some additional properties may be required. These properties, that are defined in a *functional model*, are not attributes of the part. In particular, they are not guaranteed by the *supplier* who describes the part by its *general model*. These properties are called the *representation attribute.*

EXAMPLE 4: For a ball-bearing, the inner diameter and outer diameter, etc., are attributes. The ball diameter may be described in the general model. It is an attribute of the bearing. If that diameter is not described in the general model, it can be described in the functional model that produces the geometry. In this case, it becomes a representation attribute, and can be used only to produce that representation (the parts supplier may not wish to guarantee the ball size).

• **Attribute of a part (Level 2)**: an invariable element, characteristic of a *part*, whose value can be inferred from the part *identifier* once the part is chosen, regardless of the context in which the part is inserted. The type of an attribute may be numeric, alphanumeric or Boolean. It may also be a part. The type of such an attribute is defined by the *family* (whether simple or generic) to which it belongs.

EXAMPLE 5: A supplier XYZ offers a bearing "Y bearing with cast-iron bearing plates" consisting of a body and a bearing element. The "body" and the "bearing element" are attributes of the bearing. These attributes can be represented in a Level 2 library.

• **Attribute of a part (Level 3)**: an invariable element, characteristic of a *part*, whose value can be inferred from the part *identifier* once the part is chosen, regardless of the context in which the part is inserted. The type of an attribute may be numeric, alphanumeric or Boolean. It may be a part. It may also be defined as a list of numeric, alphanumeric or Boolean values, or as a list of parts.

• **Behaviour representation attribute**: property of a *part* which depends upon some *context parameters* and which is useful for representing and facilitating the selection process inside a *parts family*. Since it is a variable, defined by the values of context parameters, a behaviour representation attribute is not an *attribute of the part*. It is represented by a *parts supplier* in a *general model class* and derives from context parameters, and possibly from parts' attributes, through a *derivation function*, or a sequence of derivation functions (see: *representation attribute*).

EXAMPLE 6: The life time of a bearing depends upon the load of the bearing. This life time may be used for bearing selection and constitutes a behaviour representation attribute of a bearing family.

• **Basic semantic unit (BSU):** identification concept for dictionary descriptions (which consist of *descriptors*), which thus provides a means to refer to these dictionary decriptions. In this International Standard, the following information units are associated with a BSU: *supplier, class, attribute,* program library, domain of values (type), *table* and document. Each basic semantic unit corresponds to an entry in the *semantic dictionary*. It carries the information about the *absolute identifier* of that entry. It is associated with a *descriptor,* and possibly, a content.

EXAMPLE 7: Each property of a part is associated with a BSU that identifies that property.

• **CAD system:** the term Computer Aided Design system (CAD) is used in this Standard to denote any kind of computer modelling system which generates and the manages product data. The product data are defined in one or several modelling spaces in which each part of the product can be positioned.

• **Class**: a set of data and programs used to provide computerised representation of a set of elements regarded as related. A class possesses two characteristics. (1) Any class is part of a set of classes organised hierarchically and implicitly possesses all the properties defined by the classes from which it descends. This mechanism is termed *inheritance*. (2) Every class possesses a mechanism termed instanciation. This mechanism allows a computer representation - termed an *instance* - to be produced from the class, for each of the elements in the set of elements represented by the class itself.
The notion of class hierarchy is used in this standard to provide computerised representation of (1) a universe of *parts* that is structured according to a hierarchy of *families (general model classes)*, (2) the universe of *functional views* of parts *(functional view classes)* and (3) the universe of all the elements required to produce those functional views (*functional model classes*). Any class possesses a *code* for identifying it, and a *version* that allows changes in it over time to be mapped.

EXAMPLE 8: The instance of a class of general models representing a parts family is a general model that provides representation, in an integrated library, of a particular part in this family.

• **Code**: a character string that has a fixed maximum length, cannot be translated and is associated with every basic semantic unit in a *supplier library*. The supplier's code is defined by part 26 of this International Standard. All the other codes are chosen by the supplier of the element and shall comply with the following properties of uniqueness:

- a code associated with a class is unique over all the classes defined by a supplier;
- a code associated with an element in a class shall be unique over all the elements of the same type for all the classes in which such an element is defined or *inherited*.

Accordingly, a concatenation of codes allows absolute identification of every *basic semantic unit* referred to in a library. A concatenation of this kind is termed the *absolute identifier* of the element.

• **Component:** a part that is not decomposable.

• **Context parameter**: a variable whose value characterises the context for the insertion of a *part*, and is necessary either to produce a given *functional view*, or to select a given *part*. The value of such a variable must be transmitted (whether directly or indirectly) by the *user* to the *library* for the view to be created.

> EXAMPLE 9: In order for a geometric functional view of a spring that will be represented in compression in a CAD model to be produced by a method contained in the library, the compressed length of the spring shall be transmitted by the user to the method.

• **Data:** a representation of facts, concepts, or instructions in a formal manner suitable for communication, interpretation, or processing by human beings or computers.

• **Data specification language:** a set of rules for defining *data* and their relationships suitable for communication, interpretation, or processing by computers.

• **Definition document**: a physical data medium (usually paper) describing a *family* (*simple* or *generic*) of *parts* (abstract or supplier's one).

• **Definition table**: a two-dimensional array used to define the set of *parts* that belong to a *simple family*. Each row of the table defines a part. Each column describes the corresponding value of an *attribute*. This table may be *virtual*.

• **Derivation function**: relationship serving to calculate the value of an attribute from the values of other attributes. This type of function reflects a relationship defined with tables or algorithms in the specification of a *parts family*. A derivation function defined in a *generic family* is *inherited* by its descendants, except when it is *overridden* by another derivation function that results in the same attribute.

• **Derived attribute**: *part attribute* whose value can be inferred from its *free attributes*, either directly by a *derivation function* whose variables are free attributes, or indirectly by a derivation function that may itself furnish certain variables that are derived attributes. Every derived attribute in a *generic family* (*generic attribute*) is necessarily derived in all its descendants, since both the attributes and the derivation functions are *inherited* (however, a derivation function can nevertheless be modified by an *override*).

> EXAMPLE 10: In the French standard for screws, NF E 25-112, the thread length is derived from the length under the head and the diameter. The derivation function (defined in a table) allows that derived attribute to be inferred from these two identification attributes. If the same derivation function was true for all the sub-families of a given family of screws, it could be factored for that *generic family* as a whole. It would then apply implicitly (inheritance) to each of the sub-families.

• **Descendant of a generic family**:Y is an (immediate) descendant family of a *family* X, if the *is_a* relationship (*inheritance*) exists between them, i.e. Y is_a X. Then a *descendant family* of (some family) A is either an immediate descendant family of A or a descendant of an immediate descendant of A.

• **Descriptor**: standardised amount of information, associated with each entry of the *semantic dictionary* that defines this entry, e.g., for an *attribute* entry: *code*, *short name*, *unit*...

> NOTE 1: The descriptors used for each parts family entry and for each attribute entry, are defined in part 42 of this International Standard with their precise structures.

• **Dialogue interface**: a number of entities designed to enable user access to the content of an *integrated library*. These entities consist of data items included in each *supplier library*.

> NOTE 2: The data items that shall be provided by library suppliers to afford user access are specified in part 42 (for the semantic dictionary level) and in part 24 (for the library content level) of this International Standard.

• **Dictionary entry**: an unit of information stored in the *semantic dictionary*.

• **EXPRESS:** the data specification language defined in ISO DIS 10303-11.

• **Family**: (see: *Family of parts, Simple family of parts, Generic family of parts*).

• **Family of parts**: a grouping of *parts* that a *parts supplier* decides to form together. A family of parts is either a *simple family of parts* or a *generic family of parts*.

• **Free attribute**: a *part attribute* that is defined or *inherited* at the level of a *generic family* of *parts*, and does not derive from other attributes of that family. The knowledge of all the free attribute values within a family is sufficient to define unequivocally a part within that family. Free attributes are chosen by the *supplier* of each parts family, and an ordered description of them is given in the specification of that family. Any attribute that is not free is called a *derived attribute*. These attributes are therefore functionally dependent upon the free attributes. The free attributes of a simple family of parts are called *identification attributes*.

EXAMPLE 11: In the (generic) family of ISO screws, the standardization committees may opt to characterize such a (generic) screw by the attributes of length and diameter attributes, both of which then constitute free attributes.

• **Functional model class**: a grouping, in the format of a *class*, of the set of *functional models* that provide information of the same *representation category* for the different *parts* of one or more *families*.

• **Functional model of a part**: an *instance* of a *functional model class* that may be (logically) created inside an *integrated library*. A functional model is always associated with a g*eneral model*, and serves to produce one or more *functional views* of the *part* represented by that general model in the user's *CAD system*.

EXAMPLE 12: A functional model of a precisely defined screw may, for example, be used to produce its different geometric views. It accesses the screw's attribute values, may contain the values of the representation attribute values if some of them are necessary, and also contains the programs (methods) which are used to produce these different geometric views. The execution of such a program creates in the CAD system the complete set of geometric data for a geometric functional view in the CAD system.

• **Functional view of a part**: an instance of a *functional view class* that may be created in a *CAD system* model by a *functional model of part*. A functional view belongs to *product data*. It refers to a *general view* and corresponds to a *representation category*.

NOTE 3: In the information model defined in part 24 of this International Standard, a functional view is defined as a SUBTYPE of an ISO 10303-43 representation.

• **Functional view class**: description, as a *class*, of each *representation category* capable of being represented in an *integrated library*. Each functional view class has a name, called the *view_logical_name*, which identifies the user perspective of the view (e.g., geometry, inertia, kinematics, etc.). Each class may also possess *view control variables* of enumerated type, which serve to specify more precisely the perspective adopted by the user (e.g., geometric_level = solid). Some functional view classes are intended to be instancied in the user *CAD System*. Such functional view classes specify the structure of their instances. Some other functional view classes are not intended to be instancied. The functional model classes referring to such functional view classes are only intended to be used during the user selection process.

EXAMPLE 13: A functional view class for geometry may possess the following:
- the name "geometry";
- four view control variables the last of which is only defined if the value of the first is "D2": geometrical_level, detail_level, variant, side;
- one (optional) view attribute: insertion_placement, and
- two (mandatory) view attributes. One, called "items", that contains geometric representation items ; the other called "context_of_items", that contains the geometric representation context in which the representation items are defined.

NOTE 4: The content of the different functional view classes are defined in the view exchange protocol series of parts of this International Standard.

• **General model class**: a grouping, in the format of a *class*, of the set of *general models* that provides computerised representations of the set of *parts* in a *family*.

• **General model of a part**: an *instance* of a *general model class* that may be (logically) created inside an *integrated library*. A general model provides computerised representation, in an *integrated library*, of each part of the *family of parts* represented by the general model class. A general model produces in the user's CAD *system* a set of data, termed a *general view*, that *identifies* the part that it represents.

• **General view of part**: an *instance* of a *class of general view* that may be created in a *CAD system* model by a part *general model*. A general view belongs to *product data*. and carries a part *identifier*.

• **Generic attribute**: *part attribute* defined at the level of a *generic family of parts*, and existing implicitly, under the name of *inherited attribute*, for any part in any *descendant family* of that generic family of parts. Any attribute existing in a generic family of parts is termed a generic attribute whether defined by the family or inherited by it, and whether it is be *free* or *derived*.

• **Generic family of parts**: a grouping of *simple* or *generic families* that a *parts supplier* decides to form from the families that he supplies, for purposes of classification and/or for factoring common *information*, and/or modelling generic components, and to which he attaches a name. A generic family of parts is associated with a set of attributes, termed *generic attributes*, that are defined, and may be used, to describe any of its descendant families (*visible attributes*). The description of a generic family may also contain:
- the list of the *generic attributes* that shall apply (possibly with a *NULL* value) for all the *parts* belonging to the generic family (*applicable attributes*).
If it is intended that the general model class that represents the generic family of parts is instancied, through *generic parts*, the description of a generic family shall also contain:
- the ordered list of the attributes to which values shall be assigned to identify a *generic part* within the generic family (*free attributes*);
- a set of *integrity constraints* expressing the logical relations linking generic attributes, that shall be verified for any part belonging to the generic family;
- and, if applicable, a set of *derivation functions*, expressed in terms of tables and/or algorithms, that enable the value of certain *generic attributes* to be derived from the values of other generic attributes;
All the properties defined in a generic family are *inherited* by the *descendants* of that family.

> NOTE 5: The methodology for structuring families of parts into simple families and generic families is defined in part 42 of this International Standard.

• **Generic part**: partially known *part* defined only by its belonging to a *generic family*. This kind of part has as attributes the *generic attributes* defined in the generic family to which it belongs, or *inherited* by it (compare with: *abstract part*).

> NOTE 6: A generic family may only be instanciated if the supplier of that family describes the extension of this class, i.e., the set of the generic parts that constitutes the content of the generic family. The resources that permit the definition of the extension of a class are specified in part 24 of this International Standard.
>
> EXAMPLE 14: The concept of generic part permits the instanciation of a "screw" within a design without caring about the simple family of screws the instance belongs to. This capability requires the availability of some functional model (e.g. simplified geometry) of such a general model.

• **Icon**: drawing that describes an object. It is a simplified representation, e.g., of a *part*, or of an item that cannot be described sufficiently verbally. In an icon the symbols for the dimensions can also be described. Icons may be represented in a *supplier library*, they are used by the *library Management System.*

• **Identification**: (see: *identifier*, *supplier designation*, *user designation*)

• **Identification attributes**: *free attributes* of a *simple family of parts*. The identification attributes of a simple family of parts are chosen by the supplier and enable each of the parts in the family to be distinguished. The set consisting of:
- the *absolute identifier* of the *simple family of parts*; and
- an ordered list of values corresponding to the list of identification attributes of the family;
therefore unequivocally identify a part of that family. All other attributes can be derived from it. This set defines the part *identifier*.

> EXAMPLE 15: AFNOR Hex_Screw: length, diameter, quality_class, type, coating.

• **Identifier**: a *data* item supplying absolute and unambiguous characterisation of a particular *part*. The content of the identifier is defined in the following way: for a part which is the sole element in the *simple family* to which it belongs, the *absolute identifier* of its family; for a *parameterised part*, the absolute identifier of the parts family and the list of values of its *identification attributes*; for a *generic part*, the absolute identifier of the generic family to which that part belongs and the list of values of the *free attributes*. An identifier provides absolute characterisation of any part both in time (for storage in a database) and in space (for the purpose of exchange between systems). When a part is created, its general view also:
- contains a *supplier designation* and a *supplier identification*;
- may contain a *user designation.*
The *general view* of a *part* may be associated (by the *is_case_of relationship*) with a general view of an *abstract part* which the parts family is case of. This general view possesses its own identifier.

NOTE 7: The different representations of an identifier, both for computer interpretation and for human interpretation are specified in part 24 of this International Standard.

• **Implementation method**: a technique used by computer systems to exchange data that is described using the *EXPRESS data specification language (ISO 10303-11)*.

• **Information:** facts, concepts, or instructions.

• **Information model:** a formal model of a bounded set of facts, concepts or instructions to meet a specified requirement.

• **Inheritance**: a mechanism defined in a hierarchical structure of sets, and automatically extending any property relating to the elements in a set to an identical property relating to the elements of each of its *descendants*. In this standard, all the properties of a *generic family of parts*: *attribute* type and name, *derivation function, integrity constraint*, etc., are inherited by its descendants.

• **Inherited attribute**: a *part attribute* whose description is given in a *generic family* that is an *ascendant* of the family to which the part belongs to. An inherited attribute cannot be renamed and may be referenced by its *code* in any family that inherits it.

• **Instance**: an instance results from the *instanciation* of a *class*. An instance contains data called *attributes*. It may contain programs called *methods*.

• **Instance attribute**: an item represented as *data* in a *class instance*. An instance attribute functions as a computerised representation of the *attributes* of the abstraction that is represented by the instance.

• **Instanciation**: a mechanism associated with a *class*. This mechanism is used to produce a computer representation, called an *instance*, of each of the elements represented by the class.

• **Integrity constraint**: a logical proposition related to the *attribute* values of a part that shall be valid for all parts belonging to some *family of parts*. A part sent to the CAD system shall meet the integrity constraints defined or *inherited* by the *family* to which it belongs.

NOTE 8: In part 24 of this International Standard integrity constraints are modelled through choice entities, that define the domain of attribute values and through exclusion constraints that exclude some t-uples from the Cartesian product of the domains.

• **Integrated library**: an operational system consisting of a *Library Management System* and a *user library*.

• **"Is_case_of" relationship**: a relationship providing a formal expression of the fact that an object conforms to the specification defined by another object. This relationship is used in this standard to associate a *family of parts* C (of *abstract* or *supplier parts*) from one *supplier* with a family of abstract parts A that belong to another supplier. It specifies in these circumstances that all the parts of C shall comply with the (partial) specification defined by A. Every element in C therefore belongs to the set of (abstract) elements defined by A (inclusion). When a family is declared a case_of another family, the *attributes* defined or inherited by the latter family may be imported, and then used, to describe the family that is a case_of. The case_of relationship defined between two families implies a relationship of the same type between the two *general model classes* that represent them, a relationship of the same type between two *instances* of these classes, and a relationship of the same type between two *general views* generated by these instances. In this International Standard the *is_case_of* relationship is only allowed between two *classes* described by two different *library suppliers* (see *is_a* relationship).

EXAMPLE 16: The choice by a standardisation body (e.g., ISO/TC184/SC4/WG2) of:
- a name for an abstract parts family (e.g. screws);
- attribute names and types for attributes that are meaningful for those parts (e.g., Thread_length, etc.);
enables any screw manufacturer who decides to declare one of its general model classes as a case_of the general model class defined by the standardisation body; (1) to import and to use to the attributes defined by the standardisation body, (2) to be accessed by the user as a case_of "screw".

• **"Is_a" relationship**: a relationship that makes possible a hierarchical structure of a set of elements based on an inclusion semantic. In this International Standard, this concept specifies the relationship between a *family* and its *ascendant generic families* in the hierarchy of families of a *parts supplier*. Every part belonging to the former family implicitly belongs to its ascendant families and therefore possesses all the properties defined for the objects belonging to them. This property is called *inheritance*. The is_a relationship defined for a hierarchy of *families* implies a relationship of the same type for the hierarchy of *general model classes* that represents it. In this International Standard the is_a relationship is only allowed
– either between *class*es described by the same *library supplier* or
– between a *class* described by a *library supplier* and a pre defined *class* described in one part of this International Standard (compare with: is_case_of *relationship*).

• **"Is_part_of" relationship**: a concept that formalizes the relationship between the constituents of a whole entity and the whole entity itself. In this International Standard, this concept specifies the relationship in an *assembled part* between a constituent *part*, represented as an *attribute*, and the assembled part. This relationship can only be represented in a *level 2* or a *level 3* library.

• **"Is_view_of" relationship**: a relationship that formalizes the concept of multiple representations of an object. In this International Standard, the is_view_of relationship between a *general model class* and a *functional model class* is used to specify:
(1) that for every *instance* of a general model class, there is a corresponding instance of this *functional model class*;
(2) that this *functional model* instance possesses *methods* allowing the construction of certain *functional views* of the *part* corresponding to that g*eneral model* instance.
The is_view_of relationship defined between *classes* implies a relationship of the same type between (*general* and *functional*) *models instances*, and a relationship of the same type between the (*general* and *functional*) *views* produced by those models.

• **Level**: a consistent set of definitions that represent a given field of study. The conceptual model of libraries defined in this International Standard has three levels. Level 1 limits the type of *attributes* to be real, alphanumeric and Boolean. It allows a library representation capable of generating sets of *parts* that are sufficiently simple for such a description to suffice. Level 2 allows an attribute to have a *general model class* as its type. It represents a library capable of managing sets of *assembled parts*. Level 3 allows an attribute to be list structured. It permit the representation of complex assemblies.

NOTE 9: The EXPRESS information models of Level 1 and Level 2 libraries are specified in part 24 of this International Standard.

• **Librarian**: see: *Library Management System*

• **Library:** an identified set of identified items which may contain or generate information required by a user (see: *parts library, user library, integrated library, supplier library*) .

• **Library end user**: the library end user is usually an operator searching for *parts* to be used for some purpose, e.g., to fulfil some function in a product. From his CAD workstation, the library end user:
– consults the *data* contained in the *library*;
– selects a given *part*;
– requests the transmission of a selected *view* of this *part* to his *CAD system*.

• **Library Management System (LMS)**: a software system that enables the *library end user* to use the content of an *integrated library*. This software system is not standardised. It is nevertheless assumed that it renders several services (example: concept of *semantic dictionary*) specified by the Standard and considered in its design.

• **Library supplier**: an organisation that describes, in a standard format, and guarantees the description of a set of *parts* characterized by their *general models* and/or of a set of *functional models* of parts. The organisation may be a *supplier* of *families of parts*. It may be a standardisation organisation that specifies families of *abstract parts*, or it may be a software supplier supplying *functional models* associated with another *library* defined by a *parts supplier*.

• **Long name**: a character string, not more than 30 characters in length, used to define the meaning of the *library* items accessible to a *library end-user*. A long name belongs to each *descriptor* of a *semantic dictionary* entry and can be translated into different languages. The *Library Management System* shall provide the functions that enable the user to conveniently consult the long names of the items he is handling in a consultation and/or selection process.

EXAMPLE 17: For the screws family: Length under head, Length up to the Head, Thread length.

• **Metadata:** data whose values specify the structure of some other data.

• **Method**: a *program* associated with an *instance* of a *class*. In a *supplier library*, methods are associated with the *functional model classes*. A method accesses a *representation transmission interface* and the instance that supports it. Like any property of a *class*, a method can be *inherited*.

• **Model**: a simplified representation or description, describing only those aspects considered to be relevant to the context of one point of view.

• **Model of a part:** an *instance* of *class* that may be (logically) created inside an *integrated library* and that constitutes a *part representation* (see: *general model of a part, functional model of a part*).

• **NULL**: a specific value added to the domain of any *attribute*. If an attribute has the NULL value, this means that the corresponding property has no value. A NULL value may only be explicitly assigned to a property in a *level 2* or *3 library*, and for properties defined as OPTIONAL. Whatever the *level* of a *library*, *NULL* values may appear in a class instance generated by the *library*. It only means that the property has no value.

• **Object oriented analysis:** an analysis of the *information* about one universe of discourse based on the *object oriented paradigm.*

• **Object oriented model:** an *information model* based on the *object oriented paradigm* .

• **Object oriented paradigm:** a paradigm which proposes to *model information* in terms of *classes.*

• **Object_name**: a value that identifies each class instance. When a product model contains part occurrences , each *part* occurrence is associated with an object_name.

• **Override**: in an *inheritance*, a mechanism whereby an inherited element is masked by an element of the same type defined in a descendant. This mechanism is used to eliminate an inherited *derivation function* for which the result is the same *attribute* as another derivation function defined in the *descendant*. If a derivation function is overridden in a *generic family*, the new function is inherited by all its descendants.

• **Parameterised part**: (see: *simple family of parts*)

• **Part**: a collection of material or functional elements that are equivalent, at a defined level of abstraction and capable of being characterised by an *identifier*. (see: *abstract part, supplier part*).

NOTE 10: In this International Standard, it is assumed that any part belongs to a *simple family of parts* of which it can be the sole element, and that the set of simple families of parts from a *parts supplier* is structured by the supplier in a tree-structured classification. In a computerised *library*, this tree-structured classification is represented by a hierarchy of *general model classes*. A part may be either an *abstract part* or a *supplier part.*

NOTE 11: The parts considered in this International Standard are the parts which are described independently of any product in which they may be used or inserted.

• **Parts library:** a *library* which either contains or may generate *information* about a set of *parts.*

• **Part occurrence**: an abstraction that represents, in the *product data* managed by a *CAD system*, a particular *part* incorporated in that product. A part occurrence is characterised by both the *identifiers* of the part to which it corresponds, and a certain position in some modelling space. Every part occurrence is assumed to be characterised by a name, termed the *object_name*, which enables it to be identified in the product model.

EXAMPLE 18: In a model for a rotating machine, the "same" XXY bearing - i.e., a bearing with the same XXY part number - may appear several times. Each of these objects is an occurrence of the same part

instance. These different objects are distinguished by a different position in the space in which the rotating machine is modelled. They are identified by different object_names.

• **Parts supplier**: an organisation responsible for specifying *abstract parts families* or for supplying *supplier parts families.* Only this organisation can be identified as the supplier of the library that contains the *general model classes* describing those parts families.

• **Product:** a thing or substance produced by a natural or artificial process.

• **Product data:** a representation of facts, concepts, or instructions about one or more *products* in a formal manner suitable for communication, interpretation, or processing by human beings or by automatic means.

• **Product information:** facts, concepts, or instructions about one or more *products*.

• **Program:** a representation of a process in a formal manner suitable for communication, interpretation, and processing by computers.

• **Property:** a term that represents either a *part attribute* or a *context parameter* or a *behaviour representation attribute* relating to a part, or a *representation attribute* relating to a representation.

• **Reference hierarchy:** a hierarchy of families of abstract parts which supplier libraries may reference and which provide a mechanism for multi-supplier access. It is intended that standardisation organisations specify these hierarchies.

• **Representation attribute**: property whose value is not defined in the *general model class*, but which is necessary to produce some *functional view* of a *part* (e.g., geometry). Representation attributes are defined in *functional model classes* of *parts*, and are used to create a certain *functional view* of these parts. Representation attributes are not included among the *attributes of a part*, as they are only defined within a particular *functional model class*. Any representation attribute is *derived* from attributes of the part with which it is associated and/or from *context parameters* through a *derivation function* or a sequence of derivation functions (see: *behaviour representation attribute*).

EXAMPLE 19: In a tap-body standard, the lower radius of the body is not defined in the standard. Since its value is required to produce a geometric view, it must be described in the form of a representation attribute in the functional model that produces the geometric view or views, or be encoded directly in the methods that generates them. In the latter case, this value can not be ascertained by the user or by the other (possible) functional models that are inheritors of the functional model that produces the geometric view. Furthermore, it cannot be evaluated automatically by a derivation function.

• **Representation category**: an abstraction used to classify the *information* about a *part* that corresponds to different *user* perspectives. In the *model* defined in this International Standard, this classification is formally expressed in terms of *view_logical_name* and in terms of *view_control_variables*.

• **Representation of a part:** a set of *data*, and possibly *programs*, that represents *information* about a *part*. In a *parts library*, part representations are described through *general* or *functional models*. In a *CAD System*, representations of parts are described through *general* or *functional views*.

• **Representation transmission interface**: a standardized interface designed to create the *information* comprised in *views* of a *part* in the *CAD system* model on the basis of descriptions contained in a *parts library*. The *library supplier* description is based on the existence of this interface in the target CAD system, and all CAD systems shall possess this interface to enable the *supplier library* to be exploited. ISO 13584-31 defines a special representation transmission interface: the programming interface used for creation by a program of geometric entities in the CAD system model. Other standardised interfaces may be used by the library supplier provided that he declares them in the exchange file.

EXAMPLE 20: A supplier of mechanical parts may choose to describe the geometric representations of his parts in the form of data models encoded in a standardised geometry exchange format (e.g., ISO 10303). Similarly, a supplier of electronic components may describe the structural representation of his components in a standard description format (e.g., VHDL, Edif). Such supplier libraries can obviously only be exploited if (1) the Library Management System recognises the interface and (2) the CAD system supports that interface.

NOTE 12: The representation transmission interface specification may consist of an external referencing mechanism that allows library information to reference product data.

• **Resource construct:** the collection of EXPRESS language entities, types, functions, rules and references that together define a valid description of data.

• **Semantic dictionary**: a table in which each entry contains a *BSU* that identifies it, and a *descriptor* that describes it. The BSU allows an *absolute identifier* to be constructed for each entry. One meaning (a "semantic") corresponds to each entry in the dictionary. Conversely, two dictionary entries identify two different semantics. This semantic identification function also enables the dictionary to provide automatic two way translation between the external names, that are user-accessible, and the internal codes used in the library. All the *basic semantic units* defined in this International Standard are entries in the dictionary.
Four formal relationships (*is_a, is_case_of, is_part_of, is_view_of*) allow the specification of semantic links between different entries of the semantic dictionary of a *user library*. For human beings (*library suppliers, library end users*, application programmers...) each entry corresponds to a precise and acknowledged meaning. The computer software uses these human-defined links to provide *services.*

> EXAMPLE 21: The detail_level view_control_variable of geometrical views corresponds to a dictionary entry. This standard assumes that the library supplier and the end-user understand the difference between the "simplified" and "extended" values. The software system may allow the user to select, if they exist either the simplified or the extended geometrical view of any part, to change from one level to another, etc.

> EXAMPLE 22: The ISO.screw.thread_diameter and the ISO.nut.thread_diameter should be two entries of the dictionary. In this case, a program may be developed that checks the consistency of a bolt assembly for any screw and nut couple whose general model classes refer to these entries.

• **Services**: functions described by this International Standard that are assumed to be supported by the *Library Management System.*

• **Short name**: a character string, not more than 15 characters in length, which constitutes an external representation of the internal *code* of a *library* item accessible to a human *user*. A short name belongs to each *descriptor* of a *semantic dictionary* entry and can be translated into different languages.

• **Simple family of parts**: a set of *parts* to which a *parts supplier* attributes a name, such that each part can be distinguished from the others by means of the values of certain *attributes*. If the set contains more than one element, the parts supplier describes the ordered lists of attributes that shall be defined to ensure *identification* of each element in the set.

• **Structure:** a set of interrelated items of any complex thing, and the relationship between them.

• **Structured part**: an *assembled part* that possesses its own *identification*, independent of the parts that constitute it, but for which the constituent parts are represented as *derived attributes*. This form of representation allows (1) the user to consult the system for information or to store it in his *CAD system* database, and (2) automatic *instanciation* of the corresponding *general models*. Such instanciation allows a *method* belonging to one *functional model* related to the structured part to trigger similar methods belonging to the functional models of each of the constituent parts.

> EXAMPLE 23: The supplier XXY offers a PDN double bearings, that consists of two bearings and one shaft. The PDN double bearings can be acquired ready-assembled from the supplier and is identified by an XXY part number. Representing the constituent bearings with an identifier comprised of derived attributes provides information to the user that is useful for subsequent maintenance. When instancing the class of XXY PDN double bearings, such a representation also allows instanciation in the library of the general models of the corresponding bearings. Thus, the methods for creating a geometrical view of the structured part mainly consist of triggering methods for producing geometrical views of each of the two bearings.

• **Supplier**: (see: *library supplier, parts supplier*).

• **Supplier designation**: a character string produced by the *Library Management System* that is associated with every *part*. When a part is selected within a *library*, this character string is assigned as the value for the pre-defined *attribute* "supplier_designation" in the *general model* for that part. When a *general view* of this *part* is created in a *CAD system*, this character string is also assigned as the value for the pre-defined *attribute* "supplier_designation" in the *general view* for that part. The content of the string can be defined by the *supplier* in the form of a string expression. When such an expression is not supplied, the Library Management System constructs a default supplier designation. The default designation consists of:
- the supplier short name;
- the short name of the class of general models to which the part belongs; and

- the list of values of the identification attributes.
When an identification attribute has a *NULL* value, its value is represented as a question mark ('?').

EXAMPLE 24:

AFNOR Hexagonal Screw 12, 60, 8.8, 1	(with no supplier expression)
Hexagonal Screw 12-60, 8.8, Type 1 NF E 25-112	(with supplier expression)
XYZ-supplier large_flat 10, 5	(with no supplier expression)
XYZ-supplier large_flat Length=10, width=5	(with supplier expression)

• **Supplier identification**: a character string produced by the *Library Management System* associated with every *part*. When a part is selected within a *library*, this character string is assigned as the value for the pre-defined *attribute* "supplier_identification" in the *general model* for that part. When a *general view* of this *part* is created in a *CAD system*, this character string is also assigned as the value for the pre-defined *attribute* "supplier_identification" in the *general view* for that part. The content of the string can be defined by the *supplier* in the form of a string expression. When such an expression is not supplied, the Library Management System constructs a default supplier identification. The default identification consists of:
- the supplier absolute identifier;
- the code and version of the class of general models to which the part belongs; and
- the list of values of the identification attributes.
The list is represented between parenthesis, the value is represented according to the ISO 10303-21 physical file format. When some identification attribute has a *NULL* value, its value is represented as a question mark ('?').

EXAMPLE 25:

FR00012-000.E_25_112-001 (12, 60, 8.8, '1')	(with no supplier expression)
NF_E_25_112 (12-60, 8.8, '1')	(with supplier expression)
FR00035-000.114-003 (10, 5)	(with no supplier expression)
XYZ-supplier.large_flat ('NLM_1140100150')	(with no supplier expression)

• **Supplier part**: a physical *part* that can exist in a large number of equivalent copies and that may be supplied by the *supplier* of the part. The *simple family of parts* that defines such an element is described by the supplier as a *class* (compare with: *abstract part*).

• **Supplier library**: set of *data*, and possibly of *programs*, that describes in the standard format defined in this International Standard, a set of *parts* and/or a set of *representations of parts*.

• **Table**: (see: *definition table, virtual definition table*)

• **User**: (see: *library end user*)

• **User designation**: a character string that can be associated, in a *user library*, with each or some of the *parts* included in that library. When a user associates such information with a part, it is automatically assigned by the *Library Management System* to the pre defined "user_designation" attribute of the *general view* of the part. This International Standard defines the (possible) existence of this attribute, but not its content.

• **User identification**: a character string that can be associated, in a *user library*, with each or some of the *parts* included in that library. When a user associates such information with a part, it is automatically assigned by the *Library Management System* to the pre defined "user_identification" attribute of the *general view* of the part. This International Standard defines the (possible) existence of this attribute, but not its content.

• **User library**: the information that results from the integration of one or more *supplier libraries* by the *Library Management System* and possibly from a adaptation by the user. The term user library therefore refers to the content of an *integrated library*.

• **Version**: a three digit number which identifies the last date of modification of a semantic dictionary entry definition.

NOTE 13: The mechanism for creating versions is specified in part 42 of this International Standard.

• **View**: the structuring unit for the *data*, produced by an *integrated library*, that outputs a *part* representation to the *CAD system* for the purpose of representing *parts*. There are two types of views: *general views*, and *functional views*.

• **View attribute**: set of *data* that is contained in a *view* and which has an *absolute identifier* associated with it. This *absolute identifier* allows the *CAD system* to interpret and to use the value of the element.

EXAMPLE 26: In a geometric functional view the list of insertion points and placements may be identified by a view attribute *absolute identifier*. It may be processed by a specific CAD system application (e.g., to link a specific insertion point with some other geometric entities).

• **View control variable**: variable of enumerated type, associated with a *view logical name* that is intended to specify the exact nature of the *representation* required by the *user* when he invokes a *view logical name*.

EXAMPLE 27: For the "geometry" view: geometric_level, detail_level, etc.

• View exchange protocol: a part of this International Standard that describes the use of *resource constructs* and of *representation transmission interfaces* that satisfy the information requirement for the exchange of one *representation category* of *parts.*

• **View logical name**: the name of an abstract *representation category* corresponding to one perspective that can be adopted by a user regarding a *part* (e.g., geometry, inertia, kinematics, etc.). Such a name, which is either standardised, or, possibly, defined by a *library supplier*, is independent of the selected part and the process used to generate the *view*. When several representations correspond to the same abstract *representation category*, *view control variables* are used to define which precise view is requested.

NOTE 14: In part 24 of this International Standard, the view logical name is represented by a **class dictionary_element**.

• **Virtual definition table**: a *definition table* defined in the form of a set of elementary tables and possibly a set of *derivation functions* associated with join rules describing the (virtual) construction of the definition table.

• **Visible property**: An *property* defined in a *family* or in any of its ascendant *generic families*. A visible property may be defined, in this family, as an *applicable property* . It may then be used to describe the parts of this family (and of any of its descendant families). A property visible in a *family* is visible, by *inheritance*, in all its *descendants.*

3.3 Abbreviations

For the purpose of this part of ISO 13584, the following abbreviations apply.

CAD: Computer Aided Design.

LMS: Library Management System.

4 Conceptual model of parts library

The concept of a (exchangeable) parts library is defined through four reference models.

4.1 The actor reference model

Although the same natural person may, in certain cases, be required to play several roles in succession (example: the end user who also wishes to enrich his library), three classes of actors involved in the use of a parts library shall be distinguished.

The *supplier*: He or she describes in a standard format, and is responsible for the correctness of the description of a set of parts or a set of part representations. This description is called the *supplier library.*

The *integrator*: He or she integrates a set of supplier libraries in an adapted software environment associated with one or more applications. The combination of this software environment and of supplier libraries is called an *integrated library*.

The *end-user*: He or she is a CAD system user who, from his CAD workstation:

- consults the information contained in the library,
- chooses a given part,
- transmits a selected representation of this part into a model held in an adapted software environment.

The library end user shall also be able to adapt each supplier library (deletion of details, addition of attributes). The result of this adaptation is called the *user library*.

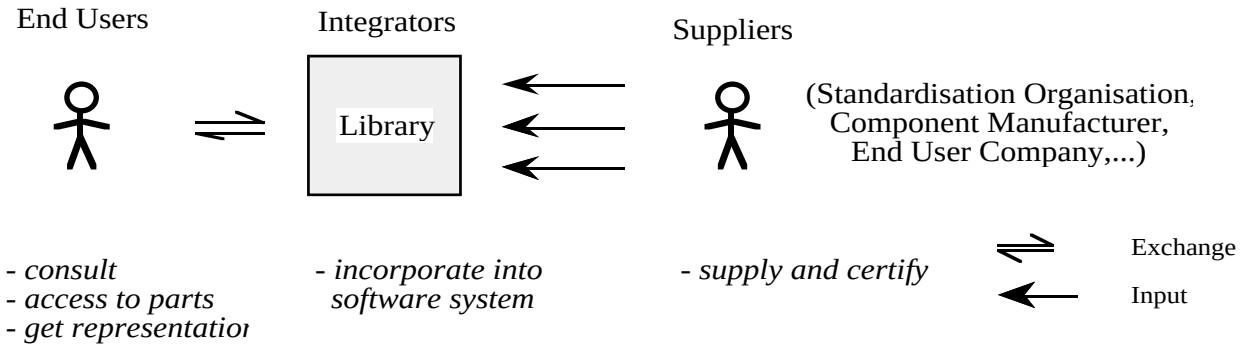


Figure 1 - Reference Model of the Roles involved in Parts Library.

In this part of ISO 13584, the openness requirement (see section 5.2) leads to a strictly limited role for the integrator. In an integrated library, all content originates from either a supplier or the user, and is identified as such. The role of the integrator is to provide a software system permitting:

1) automatic integration ("compiling") of suppliers' libraries (see fig. 2);

2) occasionally, adaptation of these libraries by the user;

3) operation of the resulting library, from the user's CAD workstation.

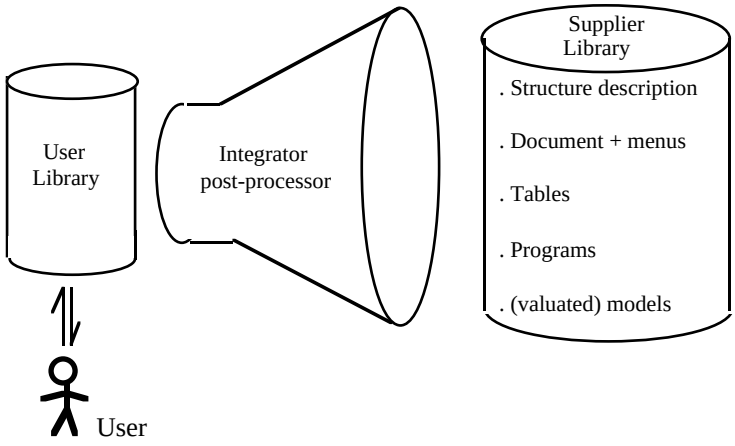


Figure 2 - Automatic Integration of a Supplier's Library

4.2 The architecture reference model

The architecture of a parts library system can be defined in terms of five sub-systems (see fig. 3).

The *dialogue interface* is a sub-system allowing the user to access , via his CAD workstation, the integrated library, and to transfer data from the integrated library to the user's screen.

The *representation transmission interface* is a set of functions, or mechanisms, assumed to be accessible from the supplier-defined representations that allows, upon the user's request, the transmission of a representation of a part to the CAD system.

The *semantic dictionary* is a table containing a set of entries that ensures the automatic translation between the external names, accessible to the user, and the internal names in the library. The following information units constitute dictionary entries: *supplier, class, attribute,* program library, domain of values (type), *table* and document.

The *semantic dictionary* performs five functions:
- it permits the expression and it ensures the storage of the relationships existing between the different items of an integrated library, whether they come from different suppliers or from the same supplier;
- it ensures the consistency of external names for elements belonging to libraries of different suppliers,
- it ensures the translation of the external names into the different languages,
- it enables a supplier to define different internal names as synonymous,
- if applicable, it contains default values for certain entries.

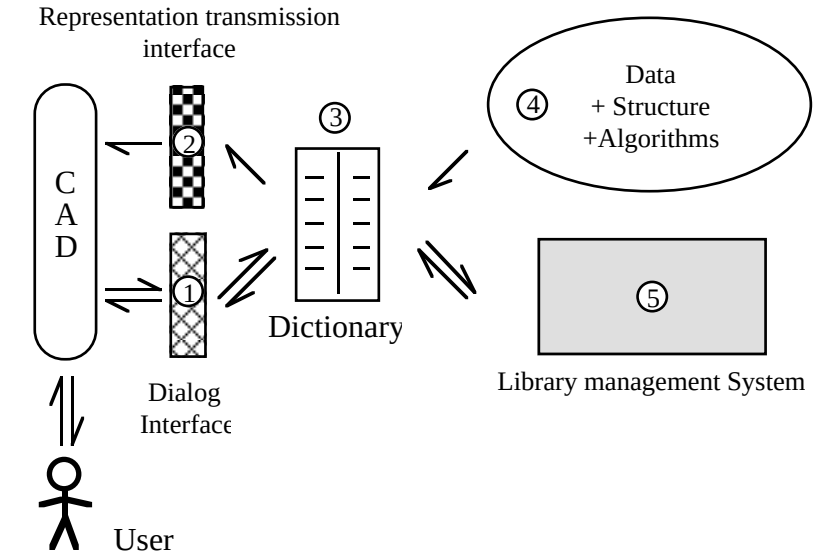


Figure 3: Reference Model of Parts Library Architecture.

The *content* of the library is the set of information making up a user library. It consists of data, and programs. Each information is originated either by a duly identified supplier, or by the user, and is identified as such.

The *Library Management System* is the software system enabling the user to use the content of an integrated library. It is developed by an integrator. This software system is not standardised. This standard is nevertheless assumed that it renders a number of services (example: concept of dictionary) specified by this International Standard and taken into account in its conception.

4.3 The information structure reference model

This model is intended to define precisely both the information to be represented in a library and the conceptual structure of this information.

4.3.1 Concept of (standard) part

The concept of *part* is the first abstraction involved in the parts library problem domain. A *part* is an abstraction used to represent a collection of material elements, or of functional items, that are equivalent, at a defined level of abstraction, and capable of being characterised by an *identifier*. There exist two different categories of parts: *supplier parts* and *abstract parts.*

A *supplier part*, or catalogue part, is a material element existing in a large number of presumably identical copies. Such an element is specified by a parts supplier who provides some identification schema to identify this abstraction. This specification is generally described in a *definition document.*

An *abstract part* is an abstract element defined by a partial specification appearing in a *definition document*, called specification (e.g., International Standard). An identification schema is always provided in this specification. The specification itself is defined by some organisation. This organisation may be considered as an (abstract) parts supplier.

Abstract part may be defined at several levels of abstraction representing a hierarchy of sets of supplier parts.

> EXAMPLE 28: A bearing with supplier_identification xxz of a bearing supplier XXX is a part that represents a set of interchangeable supplier parts. An ISO 30 BC 02 XE bearing is a more abstract part (i.e., representing a wider set). This abstract part represents the bearings supplied by different suppliers. A sphero-cylindrical bearing is a more abstract part that may be dealt with during the kinematics analysis stage of a design process (it fixes only three degrees of freedom), under the condition that its precise specification and its identifier be defined by some organisation.

A part, either a supplier part or an abstract part, may fulfil the partial specification defined by another abstract part. In that case, according to the user perspective, two different identifiers may refer to the same abstraction.

The parts are naturally apprehended through classes and class hierarchies. This is a basic method of organising knowledge for human beings. A *simple family of parts* is a set of parts that a parts supplier decides to gather and to which he attributes the same name. The supplier accordingly shall define an organised list of one or more characteristic properties, whose values serve to distinguish the different parts of the family. These properties are called the *identification attributes* of the part.

Therefore, the *identifier* of a part shall contain the following information:
- the identifier of the supplier which defines the parts family*;*
- the *name* of the *family of parts* the part belongs to,
- if applicable, the organised list of values of its identification attributes.

At some time in the design process, a part may be only partially defined. The undefined identification attributes are then unknown. In the identifier of such a part, unknown values appear in the form of *NULL values*. Such a part is said to be a partially defined part.

When a part is selected in two steps, choice of an abstract part followed by the choice of a supplier part conforming to the specification of the abstract part, the part is represented as two class instances, related by a relationship. The identifier of the abstract part, and the identifier of the supplier part conforming to the specification of the abstract part.

The relationship existing between an abstract part and one (or more) part(s) meeting the specifications of the abstract part is called an "is_case_of" relationship.

An abstract part may be linked by the is_case_of relationship with another abstract part. It permits the definition of several levels of abstraction from the user' perspective.

4.3.2 Concept of representations of a part

If a part is perfectly defined by its only identifier, such a part may have a large number of representations, e.g., solid geometry, symbolic representation, thermal behavioural model, finite-element model, and so on (see Figure 4).

Different representations may be regrouped in abstract *representation categories* identifiable by a *view logical names* (Example: Geometry, Symbol, Logic Simulation Model...). The structure of each representation category is independent, both of the chosen part and of the process (program or data) used to produce the part representation. The view logical name(short name of a class of view, see clause 4.3.11) is assumed to be accessible to the user through the dictionary.

Several representations belonging to the same abstract representation category may be accessible, in which case *view control variables* are needed to specify the desired view (Examples: Geometry_Power: solid, 3D_wireframe, 2D; Geometry_level_details: simplified, standard, detailed...).

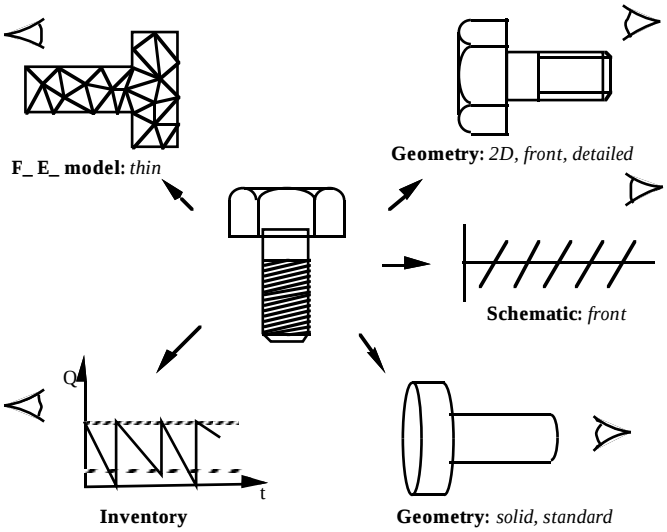


Figure 4 - The Concept of Multi-Representation

Furthermore, since certain representations depend on the context in which the part is intended to be inserted (Example: length of a jack or spring for geometric representation, temperature for service life), the expression *context parameter* applies to a variable whose value characterises the context in which a part is intended for insertion. The value of this variable shall be transmitted to the library to allow the creation of the representation.

4.3.3 Conceptual model of a part

In this International Standard, the formal description of a part is based on the object-oriented paradigm. An (object oriented) object is the representation, in the data processing universe, of an abstraction. It is characterised by:
- a state, defined by a number of attributes, accessible but not necessarily stored,
- methods, i.e. programs, that can be executed on (or with) the object.
Such methods are often activated by means of a message whose generic name is independent of the name of the activated program.

All objects having the same types of attributes and supporting the same methods constitute a *class*. Each such object is said to constitute an *instance* of the class.

For this International Standard, a part:

- *is defined by* - the class to which it belongs, called family;

- *is identified by* - the absolute identifier of its family,
  - the values of its identification attributes;

- *has* - a set of other attributes, derived from its identification attributes;

- *is linked with* - other instances which support methods (see section 4.3.9).

The expression *attribute of a part* is applied to an invariable property, characteristic of a part, whose value for each part of a family can be inferred from the specification of the family. In the simple case attributes may be of a numeric, alphanumeric or Boolean type. When representing assembled parts (see section 4.3.12), an attribute may itself be a part. The identification attributes are special case of attributes.

4.3.4 Simple family of parts

For a simple family of parts, the set of parts which constitute the family is always described using a table (virtual or real) such that:
– each line corresponds to a different part,
– each identification attribute is represented in a single column,
– other columns, if any, are used to define the concomitant variations of certain other attributes, which derive from the *identification attributes*

Such a table is called the *definition table* of the parts family (see Figure 5).

Since this table may be extremely voluminous in certain cases, it is frequently described in the form of:

- a set of basic tables,
- a set of expressions, that defines the values of some attributes as the result of functions whose domains are defined by other attributes (derivation function, see clause 4.3.5)
- and a set of join-rules enabling the (virtual) construction of a single table from the set of basic tables and of expression-defined derivation functions.

In this case, the *definition table* is said to be *virtual* (see Figure 6).

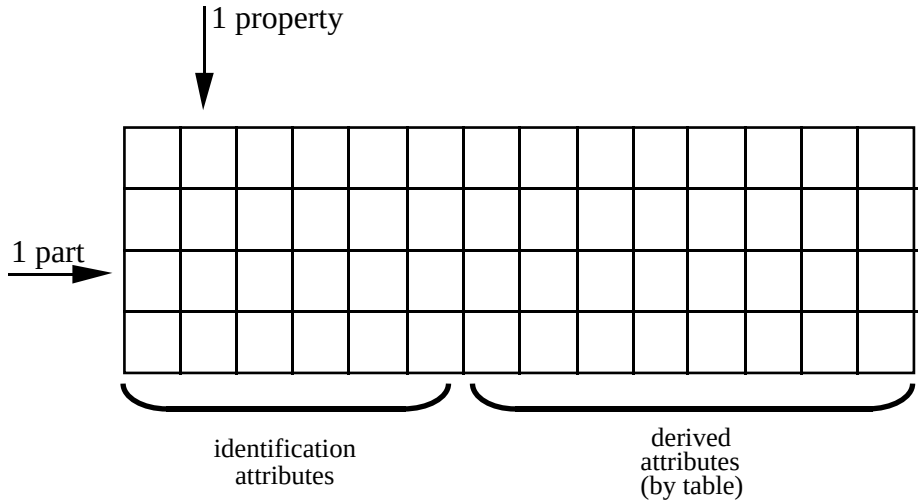


Figure 5 - Definition Table of a Simple family of parts

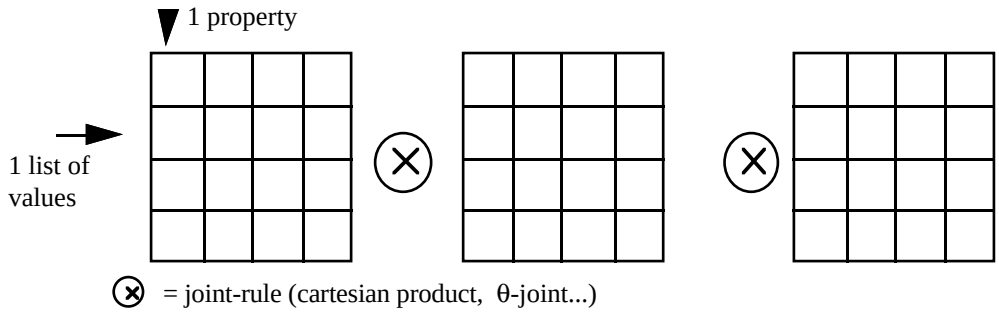


Figure 6- Virtual Definition Table of a Simple family of parts (Virtually expanded to an actual definition table).

Therefore, there are two aspects in the formal description of a simple family of parts:

1) The description of the structure of the class: attributes, methods, messages and rules. Such a description is easily worked out in an object oriented formalism.

2) The description of the definition table associated with the class. Such a description may be accomplished by means of joint-rules referring to basic tables and/or derivation functions. As the usual object-oriented languages are not capable of describing such a construction, specific resources shall be defined to express it.

NOTE 15: These resources are defined in part 24 of this International Standard.

4.3.5 Derived attributes

The specification of a simple family of parts may also contain other tables or algorithms serving to derive certain attributes, directly or indirectly, from the identification attributes. Such attributes are said to be *derived* by table or by algorithm.

The mapping that accordingly serves to calculate the value of an attribute from the values of other attributes is called the *derivation function.*

4.3.6 Exclusion constraints

It is sometimes difficult to construct a definition table that contains only the actual existing parts. It is then simpler to construct a larger table, and to add to it some constraints limiting the lines of the table which are actually authorised.

The term *exclusion constraint* is applied to a logic proposal concerning the attribute values of a part and must be valid for each authorised part.

4.3.7 Conceptual model of a structured set of parts families

Simple families of parts may themselves be grouped. The creation of such a group factors out certain properties that are generically defined for each of the simple families included in the group, for example, when the same attribute shall be defined for all the families making up the group, or when the same derivation function exists between common attributes.

The expression *generic family of parts* applies to a set of parts defined by:
- a name;
- a set of families, simple or generic, whose grouping constitutes the generic family;
- if applicable, a set of attributes, called *generic*, which are defined for any part belonging to the generic family and may be referenced to in any descendant family (*visible attributes*);
- if applicable, the list of the *generic attributes* that shall apply (possibly with a *NULL* value) for all the *parts* belonging to the generic family (*applicable attributes*).

If the supplier provides functionality that enables the library user to create instances of a generic family of parts, he or she shall also specify:
- the ordered list of the attributes to which values shall be assigned to identify a *generic part* within the generic family (*free attributes*);
- a set of integrity constraints expressing the logic relationships between generic attributes which shall be verified for each generic part belonging to the generic family of parts.
- if applicable, a set of derivation functions, expressed in terms of tables and/or algorithms for deriving the value of certain generic attributes from the values of other generic attributes;

> NOTE 16: Free attributes, integrity constraints and derivation functions shall only be provided when they are intended to create generic parts (see section 4.3.8) belonging to the generic family of parts.

The generic attributes of a generic family of parts that may be inferred from other generic attributes of this family are called *derived attributes*. Generic attributes which cannot be inferred from other attributes are called *free attributes*.

Since families of parts can be grouped into generic families of parts at several levels, a family is said to be a *descendant* of a generic family if it forms part of the families making it up, or is a descendant of one of them. A generic family of parts is said to be an *ascendant* of each of its descendants.

The properties defined at the level of a generic family of parts, including visible and applicable attributes, derivation functions and integrity constraints, are considered as implicitly defined for each of its descendant families. These properties are said to be *inherited*. This inheritance mechanism for derivation functions is illustrated in figure 7.

The relationship existing between a family and each of its ascendant families is called an *"is_a" relationship*: Any part belonging to a parts family implicitly belongs to all its ascendant families.

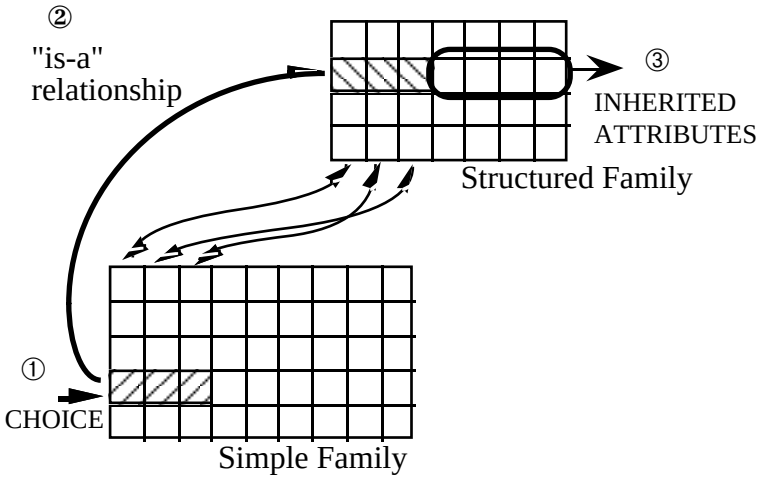


Figure 7 - The Inheritance Mechanism: inherited attributes are implicitly defined and derived.

Formal representation of inheritance is straightforward in an object-oriented formalism through class hierarchies. Therefore, the description of a supplier library in the form of a class hierarchy permits all the elements (attributes, constraints or methods) that are applicable to be factored out, in a generic way, for all sub-families.

This object oriented reference model for the generic set of families of a parts is illustrated in figure 8, using the COAD and YOURDON formalism (see: Annex C).

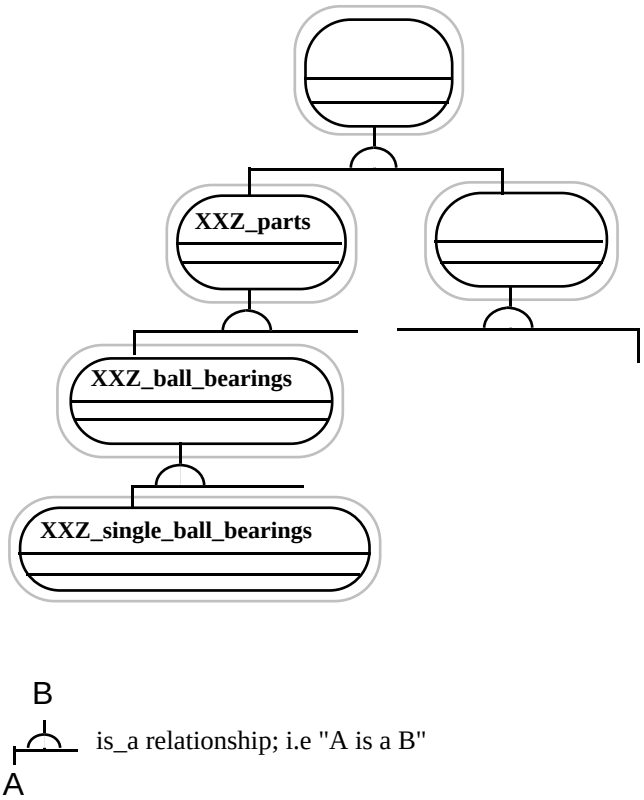


Figure 8 - Object Oriented Reference Model for Parts Families

EXAMPLE 29: In several Standards, a screw has a quality class. Certain mechanical properties depend exclusively on the quality class. If a "screw" generic family is defined as consisting of all the screw simple families defined by the standardisation organisation, and if this generic family of parts possesses, as a free attribute, the quality class, and, as derived attributes, the different mechanical properties, with the derivation function described by a table, in this case:

- every screw of a certain type (implicitly) possesses a quality class;
- every screw of a certain type (implicitly) possesses various mechanical properties;
- the values of these mechanical properties are perfectly fixed once the quality class of the screw is known (the derivation function is implicitly valid).

4.3.8 Generic part

During design, it is often useful to consider a part at the level of a generic family of parts without knowing the simple family of parts from which the part will ultimately be selected. The description for a part at this level is called a *generic part.*

Its attributes are the generic attributes specified for the generic family of parts or inherited by it. The identifier of a generic part consists of:
- the absolute identifier of the generic family of parts;
- if applicable, the list of values of the free generic attributes.

NOTE 17: If a common representation exists for all the parts belonging to a generic family of parts (e.g., schematic representation), the description of this representation can be factored to the level of this generic family. Hence, it can hence be used to represent a generic part.

4.3.9 General model and functional model of a part

A document that defines a part, whether a standard document or a manufacturer's catalogue, only describes certain properties of the part. Representation of other properties (attributes, rules or methods) is very often necessary. This may include management data specific to the company, or functional properties corresponding to perspectives not provided in the definition document (example: creation of a wire frame geometry, computation of the temperature in certain environmental conditions).

These representations correspond to several perspectives of the same object. They shall be:
– separated, to enable the supplier of each representation to certify authorship,
– logically linked to provide, for each part, all representations available in the library.

Each of these perspectives can be represented in the form of a class, i.e. by attributes and methods. The class that corresponds to the specification of the parts family defines the parts and describes their intrinsic properties. It constitutes a *general model class* of the parts family. Other classes are only meaningful with respect to this definition, i.e. to this general model class. These classes constitute *functional model classes* that can be associated with the general model classes.

The relationship existing between classes describing two models of the same part is different from the concept of inheritance. It is not a set to set mapping of inclusion of classes ('is_a' relationship), but an instance to instance relationship, in which each instance of a class is associated with one instance of the other class. This relationship is an instance connection. The identifier of the part remains unchanged, because the part is the same. Only the represented properties (attributes and methods) are enriched by the mapping. This mapping is called the "is_view_of" relationship.

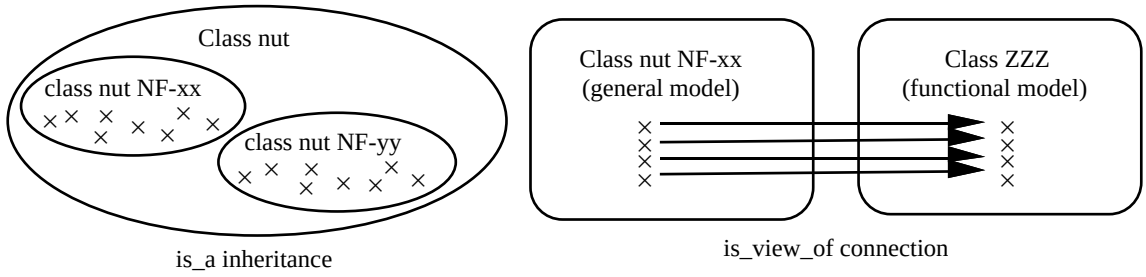


Figure 9 - The is_a and is_view_of Relationships

4.3.10 Conceptual model of a multi-representation parts library

In this International Standard part specifications and part representations are modelled, in a library, through parallel class hierarchies.

The expression *general model class* applies to the representation (in the form of a class) of the properties appearing in a parts family specification.

EXAMPLE 30: If some standard washers family specifies three properties of washers (hole_diameter, outer_diameter, thickness) together with the set of allowed values, then the general model class that models this family:
– specifies the three part attributes through three descriptors,
– describes the (virtual) definition table that identifies the allowed values and
– specifies the identification schema of a washer that belongs to this family.

The expression *functional model class* applies to the representation of the properties that belong to some *representation category* for all the parts of some parts family.

The description of a functional model class is similar to the description of a parts family, except that it contains only context parameters, representation attributes, derivation functions (if any) and methods. The methods are associated with the view logical names, and possibly with the view control variables, which allow the user to request them. The role of these methods is discussed in the next section.

The attributes of the general model of a part may be accessed by any functional model associated with it. Certain properties, which are specific to a functional model, shall be defined in its functional model class and derived, by table or by algorithm, from the attributes of the general model and the possible context parameters. These properties are called *representation attributes.*

EXAMPLE 31: In the example above, the geometry functional model:
– specifies an additional characteristics that is needed for creating the geometric representation of the washer, namely the fillet on both sides of the hole; this additional characteristic is a representation attribute,
– specifies the derivation functions that derive the value of this representation attribute out of the thickness of the washer and
– contains, e. g., in the format of a parametric program, the method that generates the solid geometry view of the washer.

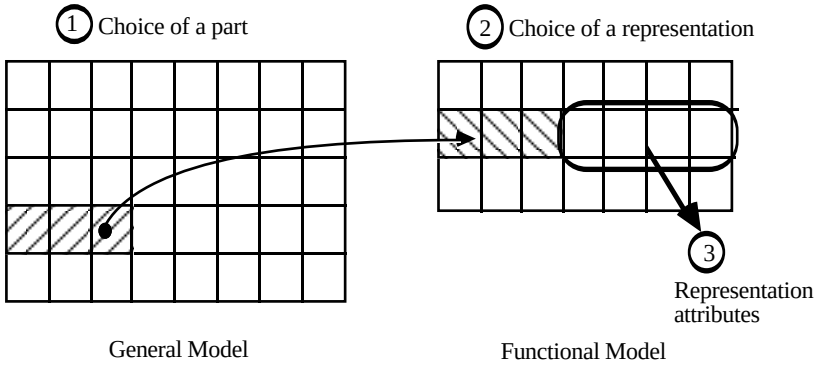


Figure 10 - Functional Model to General Model Mapping: new attributes (or methods) are accessible whenever a representation is chosen.

Various functional model classes, corresponding to different representation categories, may be related with the same class of general model, corresponding to the same parts family. On the other hand, a given class of functional models may be associated with several general model classes, possibly belonging to different supplier libraries.

Since functional models are themselves defined in the form of a class, the inheritance mechanism may be used to factorises the common properties of several different classes. The properties that are factored out may differ for each type of model.

EXAMPLE 32: If the simplified geometry representations of different simple families of screws are identical, the method that generates these representations may be factored to a superclass of all the functional model classes that describe the geometry representations of the various simple families of screws.

EXAMPLE 33: If the supplier provides functional model classes that define the purchasing conditions of the different classes of parts he or she supplies, the different representation attributes, used in these functional model classes (e. g., price, quantity of order), shall be defined in a common superclass. They are therefore inherited in all the classes that describe the purchasing conditions of the various families of parts.

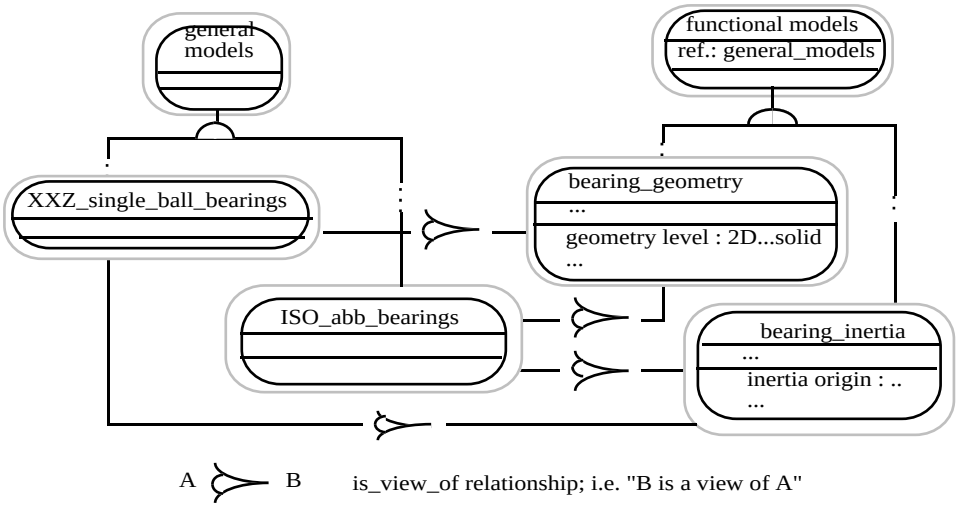


Figure 11 - Reference Model for Multi-Representation Parts Libraries: Parallel hierarchies of general model and functional model classes

The *is_view_of* relationship shall be used by the library management system, to present a multi-representation view of the part to the end user. The choice of a part is made from the class corresponding to its general model. When the part is chosen, several representations may be requested, each functional model class eventually adding its own attributes and/or methods.

NOTE 18: The selection of a part always consists of creating an instance of a general model class, nevertheless a functional model class (e. g., purchasing conditions) may be used (together with a general model class) during the selection process.

EXAMPLE 34: A library management system may (automatically) generate SQL views that contain both the part attributes (general model) and the price of each part (representation attributes of some functional model class) to support the user selection process.

4.3.11 Parts' models and parts' views

The functional model classes permit the conceptual representation of the different representation categories of a part in an integrated library.

The user requirement is to get a part representation in the product data model. This representation generally has a very different structure from the functional model. For instance, the geometry functional model may consist of a parametric program. The part representation in a product data model may be a set of geometrical entities (or an external reference to a part's parametric model together with some specific insertion characteristic data). This latter part representation is called a part's *view.*

EXAMPLE 35: In example 31, the solid geometry view of a washer (inside the CAD system) may be the B-rep representation of a circular sweep solid. This functional view results from triggering the parametric program that belongs to the functional model class.

The view is the structuring unit for the data produced by an integrated library for output to the CAD system for the purpose of representing parts. There are two kinds of views:
- the *general view* represents a part occurrence in the product data, and carries the part identifier;
- the *functional views* are the information models of the different representation categories of a part in a product data model.

The structure of a general view and the structure of the functional view which corresponds to each representation category shall also be defined by classes. These classes are called *general view class* and *functional view classes*

Views are created by a models' methods. The general views are created by a pre-defined method associated with each general model class by the integrator software system. The functional views are created by methods supplied by library suppliers in their functional model classes. A view is created by a model’s method by means of some representation transmission interface.

Each functional view:

- refers to a general view;
- corresponds to a logical view name, and possibly, to view control variable values;
- contains a set of view attributes.

NOTE 19: In part 24 of this International Standard, a part functional view is mapped onto an ISO 10303-43 representation. A part functional model, that supports methods, is mapped onto specific resources defined in this part.

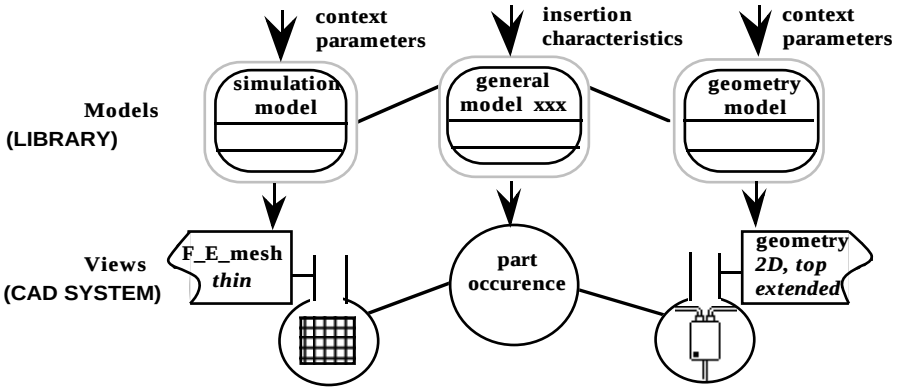


Figure 12 - Reference Model for Parts' Models and Parts' Views

4.3.12 Assembled parts

Certain parts are themselves defined as an assembly of other parts belonging to clearly defined families. These parts are called *assembled parts*. An assembled part is characterised both by the parts making it up and by a number of other characteristics which define the way in which these parts are assembled.

The conceptual object oriented model defined for the description of a set of parts can be used to describe a set of assembled parts provided the definition of the concept of an attribute is extended.

The difficulties of accounting for this new definition in certain computer environments have led to associating it with a concept of *level*. A level 1 library supports only the concept of attribute defined thus far. A level 2 library supports the description of assemblies. A level 3 library permits the use of list-structured attribute which enables modelling more complex assemblies.

In a level 1 library, the type a part attribute may be numeric, alphanumeric or Boolean.

In a level 2 library, a part attribute may itself be a part. The type of such an attribute is then defined by the family (simple or generic) to which it belongs.

In a level 3 library, a part attribute may also be list-structured. Higher level assembled parts may therefore be modelled in such libraries.

In an exchange, between the CAD system and the library, of an attribute whose type is defined by a parts family , the value of this attribute is represented by the class instance which constitutes the value of the attribute.

The whole-part relationship between a part, and the part of which it constitutes an attribute, is called an "*is_part_of*" relationship. Like non assembled parts, called *components*, functional model classes may be associated with general model classes of assembled parts (see figure 13).

NOTE 20: The threefold definition of the concept of attribute actually defines three conceptual library models. A level 1 library, easier to implement, especially in a relational environment, deals with sets of components. Level 2 and Level 3 libraries make it possible to treat an assembled part as a part.

NOTE 21: A part that consists of different solid bodies (e.g. a ball bearing) but whose attributes do not refer to these items, is not an assembled part but a component and it may be described in a level 1 library.

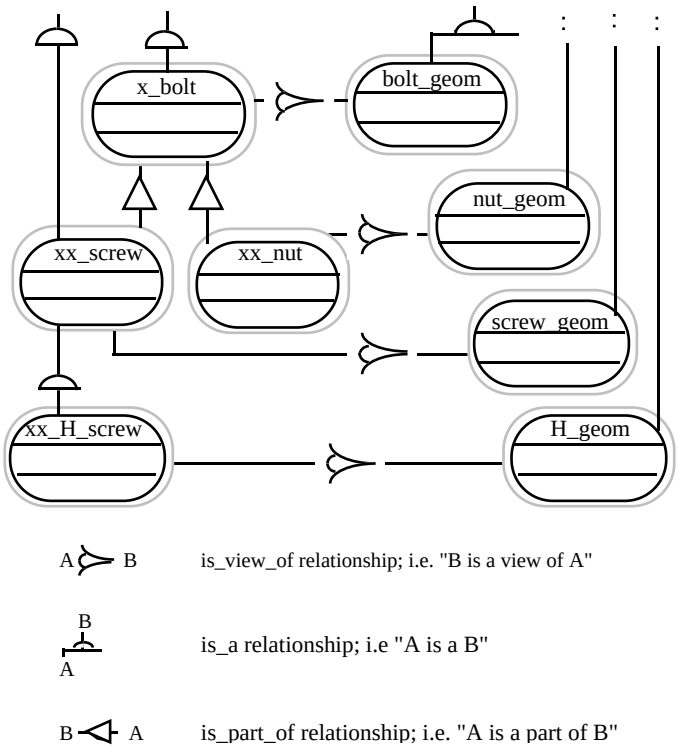


Figure 13 - Reference Model for Whole-Part Structure in Parts Libraries:

4.4 The information semantic reference model

4.4.1 Semantic description of a parts library

A part is not only a set of attribute values. It is an understandable abstraction. Each part attribute value also refers to some abstraction. These abstractions shall be defined for the parts library end-user to allow him to consult the library data, and to select parts and part representations.

In the textual description of parts families, whether in Standard documents or in suppliers' catalogues, parts families are described at two levels of abstraction.

- The first level describes the concepts about the parts family and the part attributes. Each abstraction is unformally or formally defined. It constitutes the *semantic description.*

- The second level describes the set of parts which belongs to the family. It constitutes the *logical description*.

The semantic description specifies the set of parts belonging to the parts family by intention: it provides the information to decide whether a part may belong to the parts family or not. The whole set of parts belonging to the parts family is possibly unknown.

The logical description specifies the set of parts belonging to the parts family by extension: it enumerates, in some way, all the parts that belong to the parts family.

There are some degrees of independence between both levels of description. First, some parts families have only a semantic description. An example of this case consists of the (abstract) parts families described in the so-called terminology Standards. Second, the logical description may change without changing the semantic description. This occurs, for example, when a Standard or a catalogue is updated by adding or withdrawing some parts of a parts family.

This dual description is the key which permits referencing between libraries of different suppliers. In the semantic description of its parts family, a supplier may refer to the semantic description of another parts family, specified by another supplier, generally a standardisation organisation which describes families of abstract parts

4.4.2 Conceptual model of multi-supplier parts library

In this International Standard, each parts family is described at two levels of abstraction.

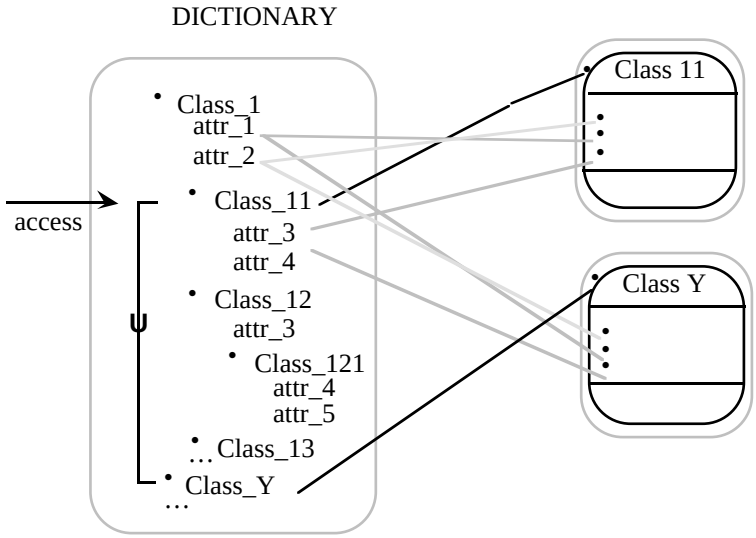
The semantic description provides the information about the parts family name and meaning, and about the part attribute names, domains of values (types) and meanings. Both the parts family name and the part attribute names are associated with a *code*, which identifies the abstraction, with a *version* which characterises updates, and with a *descriptor*, which defines the abstraction. They shall be stored as entries in a table, called the *semantic dictionary*.

The logical description, which may be missing, provides the information about the content of a parts family. Each logical description refers to an entry of the semantic dictionary, and each attribute that describes its content refers to an attribute entry of the semantic dictionary.

NOTE 22: In this International Standard, the semantic descriptions are specified by ISO 13584-42 (dictionary-schema) and by ISO 13584-24 (ISO_13584_dictionary_schema). A semantic description of a parts family is mapped onto a Component_Class. The logical descriptions are specified by ISO 13584_24 (library_content_schema). A logical description of a parts family is mapped onto a Component_Class_Content.

The same conceptual object oriented model applies to both the semantic and the logical description. Parts families are structured in a hierarchy with simple inheritance.

For the purpose of this International Standard, parts family entries and attribute entries shall be defined simultaneously by a parts supplier. This improves their definition. The composition of a family becomes clearer through the attributes entries which corresponds to this family. The meaning of an attribute is explained by the parts family defining its fields of application. The methodology for defining parts families and attributes of these parts families is specified in part 42 of this International Standard



B ⊃ A is_case_of relationship; i.e. "A is case of B"

Figure 14 - Reference Model for a Multi-Supplier Library: The semantic dictionary permits reference to pre-existing entries.

In the semantic description of a parts family, the supplier may specify that this parts family library is included in a family of abstract parts described by another supplier. This relationship constitutes the "is_case_of" relationship. In this case, the attribute entries defined for the latter may be referenced in the description of the former.

One meaning corresponds to each entry in the semantic dictionary. Conversely, two different dictionary entries identify two different semantics.

The semantic dictionary may be extended by any library supplier. A supplier may choose either to introduce new entries or to reference entries from other suppliers.

The semantic dictionary provides for multi-supplier selection and consistency. It also provides a formal model for the computerised representation of product Standards.

4.5 Access method

Three access methods have been identified to characterise the minimal information retrieval mechanisms that shall be provided by a library supplier. Other access mechanisms may be provided by the library management system implementers (e.g., customisable hierarchical access) on a system specific basis.

Hierarchical access provides functionality for accessing a family, or a part, by following pre defined, or customisable, selection trees. At each node, this access requires:
- the display or one (or more) menu(s), that contain textual and/or graphical parts family presentations;
- the family descriptions intended to provide guide lines to the user in its selection process.
The corresponding data shall be provided by library suppliers.

NOTE 23: All the dialogue resources to enable user access as specified in part 24 of this International Standard.

Direct access enables the access of a family, or a part, from an external identifier of this family, or of this part. This access requires the existence of a human readable identifier for every part and every family.

NOTE 24: The different external presentations of parts family names and attribute names are specified in part 42 of this International Standard.

Relational access enables the selection of a set of parts from Boolean expressions (predicate) concerning the value of their attributes. This access requires the existence of a data dictionary of family names and attributes.

5 Need for standardisation

The four reference models presented in the previous section now allow the precise definition of both the area that shall be covered by the Parts Library International Standard and the requirements it shall respect.

5.1 Scope of standardisation

The following shall be covered by the Parts library International Standard.

(1) The structure of an identifier which serves to identify all parts, supplier or abstract, subject of a simple or parametrized definition, irrespective of their origin.

(2) Some representation transmission interfaces (at the minimum for geometrical representation).

(3) The supplier data required by the dialogue interface.

(4) The dictionary entries corresponding to:

- the attributes that apply to any part (e. g., the mass), and progressively some specific attributes of each family;
- the view logical names of some representation categories (at the minimum for geometry);
- some view control variables (at the minimum for geometry).

(5) A supplier library description format, that provides up-dating mechanisms.

(6) Certain elements of the geometrical representations generated.

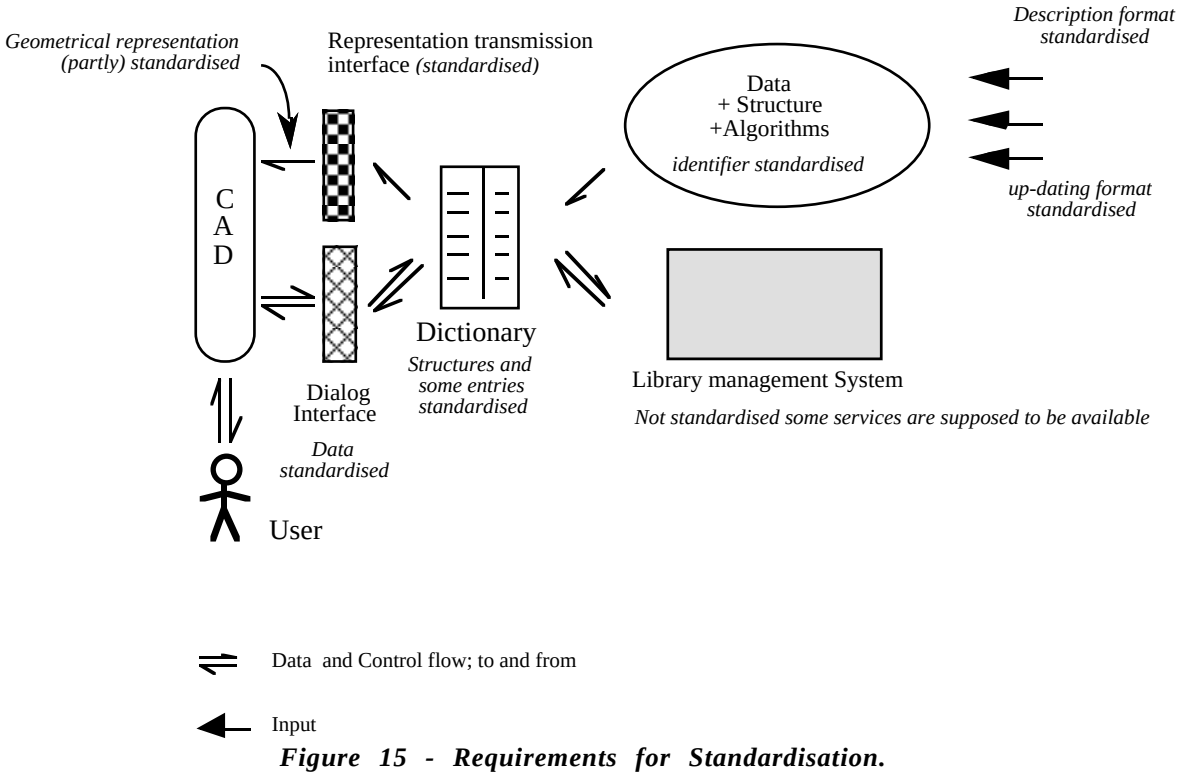


Figure 15 - Requirements for Standardisation.

5.2 Design principles

The following design principles shall be followed in the design of the Parts Library International Standard.

(1) *Minimally*. Only items whose non-standardisation would prevent the appearance of portable libraries should be objects of standardisation.

(2) *Openness*. The Parts Library Standard should be kept open at three levels:

- Level of actors: It shall assure the mutual independence of suppliers and integrators.
- Level of technical solutions: It shall be compatible with all the different technical approaches already in use or foreseeable (centralised and decentralised libraries, linked to the CAD system or separated from it, whose parametric programs are written in compiled language, in interpreted language, or in the form of parametric or variational models that can be evaluated).
- Level of representation categories: The approach used to deal with geometric representations of parts or part families shall permit suppliers or standardisation committees to propose other functional models and representation categories.

(3) *Compatibility*. The standard shall be consistent with the existing standards which deal with information about parts, particularly with ISO 10303.

5.3 Functional specification

5.3.1 Neutrality of programs

The description format shall allow the description of neutral programs. It shall be possible to deliver these programs:

- in compiled language both in source or compiled version;
- in the specific language of a CAD system;
- in the form of parametric models, if available.

5.3.2 Self-sufficiency of the description of a supplier library

The description format shall be sufficiently complete for automatic integration to be feasible.

5.3.3 Expressive power of the description format

The description format of a supplier library shall be able to describe:

1) the semantic description of a hierarchical structure of families of parts and the semantic description of their properties ;

2) a set of documents containing textual data, graphical representations and tabular layouts,

3) the logical description of a set of simple families of parts, defined as classes, and intended to be intancied;

4) the logical description of hierarchical structure of families of parts, defined as a class hierarchy with inheritance,

5) a coherent set of functional models defined in terms of a hierarchy of classes. Such classes may refer either to parts families described in the same supplier's library, or to other supplier's families presumed to be pre-existing in the integrated library.

5.3.4 Standardisation of the dictionary

All the entries appearing in the dictionary shall be freely extensible by each supplier to ensure the openness of the library.

The Standard shall define:
- the format for the introduction of data into the dictionary;
- progressively certain abstract parts families names (reference hierarchies);
- certain general purpose attributes, and then progressively the specific attributes of each family;
- the logical view name(s) of geometric representation(s);
- the view control variables of geometric representation(s).

5.3.5 Representation transmission interface

The first interface for geometry shall be defined logically in the form of a procedural interface. A FORTRAN binding shall be defined in the first release of this International Standard.

This interface shall allow the transmission of a 2D, 3D wire frame and 3D solid geometry, at least in simple form.

A procedure shall be specified to allow the use of other interfaces which may be defined by other Standards, or by other parts of this International Standard.

5.3.6 Dialogue interface

The dialogue interface consists of two parts:
– the dialogue data that shall be provided by the library supplier and hence that shall be standardised ;
– the dialogue software that shall be part of the library management system and hence that shall be logically defined.

The standardised dialogue data shall be sufficient to permit the library management system to provide the library end-user with the three kinds of access methods identified in section 4.5.

The minimal services provided shall contain the following:
– general information functionality; functions to retrieve information about parts and part families stored in the library (thus information about the general model classes),
– functional information functionality; functions to retrieve information about available representations of parts and part families (thus information about the functional model classes),
– browsing and selection functionality; functions to browse through the hierarchy of the general model classes and to select parts; the latter shall contain functionality trigger the methods for inserting a selected part, and selected part representations into the design model.
– translation functionality; functions that permit the user to query the library in its native language and that present the library internal content in the user language.

5.3.7 Convergence with the ISO 10303 (STEP) Standard

The Parts Library Standard shall be interoperable with ISO 10303. It shall be possible:

- to create parts representations in a product data model conforming to ISO 10303 Application Protocols from a library conforming to this International Standard;

- to refer to a part defined in a library conforming to this International Standardfrom from a product data model conforming to an ISO 10303 Application Protocol.

Annex A
(informative)

REQUIREMENTS FOR A PARTS LIBRARY STANDARD

This annex results from an analysis of end-user requirements. It presents both the requirements of library end-users, and the situation in the absence of standardisation. This should provide a better understanding of the goal of this International Standard.

A.1 Functional needs

1- Ergonomically designed, uniform access, by several access methods, to pre-existing objects of various origins.

2- In the product being designed at the level of the CAD system, integration of a part materialised by a selected representation.

3- Capability to enrich a library by adding to it:
- supplementary representations of existing objects;
- supplementary objects;
- supplementary selection criteria.

4- Capability to simplify a library by deleting:
- unused representations;
- unused families of objects;
- subsets of objects belonging to the same family which shall not be used.

5- Exchange product data containing library parts with other users.

6- Access, both by program and by interactive interrogation, to all the data available on the parts for applications other than computer aided design of products.

A.2 Economic needs

1- Economy of cost, both for acquisition and maintenance.

2- Economy of storage space.

3- Economy of machine load.

A.3 Need for reliability

1- Possibility of guarantee by each library supplier, despite integration.

2- Possibility of certification by standardisation organisations despite integration.

A.4 Situation in the absence of standardisation

The libraries are:
- incomplete;
- not harmonised from one supplier to another;
- insufficient in the available representations;
- difficult to reduce or to enrich;
- difficult to connect to the company management system;
- insufficient in selection mode;
- costly for creation and updating;
- often too redundant in data storage;
- unreliable because of the absence of any guarantee by the suppliers, or of certification by the standardisation organisations.

Annex B
(informative)

ADAPTATION OF SUPPLIER LIBRARIES BY END USERS

This annex proposes a method enabling the user to adapt supplier libraries.

A distinction must be drawn between two types of adaptation needs between two categories of users.

B.1 Large companies internal standards

In a large company, the set of internal standards is constructed, before any integration with a CAD system, on the basis of various standards and supplier catalogues. These supplier libraries are merged, modified and presented in a uniform manner.

Since the company's standardisation department is responsible for the library of internal standards, it must be considered as a library supplier. Based on the contents of various outside supplier libraries, it constructs a new library intended to be used on different CAD systems. This library shall be described in the standardised format specified in this International Standard for supplier libraries. The software tools required to carry out this work are of the same type as those used by any library supplier.

Figure 16: Building up of a Company Specific Library.

B.2 End-user adaptation of an integrated library

The problem is different here. It involves enabling an end user to adapt a supplier library, after its integration, using an integrator software, in the environment of a specific CAD system. For instance, he may wish to delete certain classes of parts, delete certain parts of a family (of which the use may be prohibited, for example) or add certain attributes specific to the company (for example, an internal part number), while preserving the original data and maintaining a reasonable volume.

This matter is especially difficult to deal with because, if the representation of the definition data of a class can, in a supplier library, be condensed in the form of several tables associated with join rules, such a representation is no longer usable if the user wishes to preserve only some of the elements and/or associate specific attributes with them (example: user_id).

No conceptual mechanism appears to be available today. It is therefore proposed that the conception of the present International Standard should rely on the following mechanisms (services assumed to be rendered by the software supplied by each integrator).

The set of tables making up the supplier library is unchangeable. The definition table associated with a parts family can be defined therein in the form of a join of several separate tables.

During the initialisation of a supplier library by a user, the user enjoys the possibility of requesting the duplication of certain tables, via the software supplied by the integrator. These tables can then be adapted by the user as follows:

1. Table lines can be deleted (Example: quality_class or coating of bolts and nuts not used in the company).

2. New attributes can be added.

3. For tables defined by join rules, the effective creation of the junction may be requested, the resulting table being susceptible to changes 1 and 2.

The tables thus modified are then processed by the software instead of the initial tables. The supplier library, thus modified, constitutes the user library.

In addition:

- the supplier library remains accessible (for example, on magnetic tape) to create a new version of the user library if required;

- during an update of the supplier library, the reconstruction of the user library may be automatic, cases of incompatibility being indicated by a warning message (Example: existence of a nomenclature for a set of values of identification attributes no longer existing in the new supplier version).

Annex C
(informative)

BIBLIOGRAPHY

[1] P. Coad, E. Yourdon: Object-Oriented Analysis, Prentice-Hall (1992).